Departamento de Ingeniería Mecánica

Ingeniería Técnica Industrial, especialidad Mecánica

PROYECTO FIN DE CARRERA

IDENTIFICACIÓN DE FISURAS SEMIELÍPTICAS EN EJES SOMETIDOS A SOLICITACIONES COMBINADAS DE FLEXIÓN Y TRACCIÓN MEDIANTE LA APLICACIÓN DE REDES NEURONALES

Autora: Alexandra Feito Redruello

Tutora: Mª Belén Muñoz Abella

Leganés, Octubre 2011

Título: Identificación de fisuras semielípticas en ejes sometidos a solicitaciones combinadas de flexión y tracción mediante la aplicación de redes neuronales.

Autora: Alexandra Feito Redruello

Tutora: Mª Belén Muñoz Abella

EL TRIBUNAL

Presidente: Alejandro Quesada González

Vocal: Álvaro Olmedo Marcos

Secretario: Patricia Rubio Herrero

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

En primer lugar quiero darle las gracias a mi tutora, Mª Belén Muñoz Abella por haberme dado la oportunidad de realizar este proyecto y por haberme ayudado y aconsejado durante la realización del mismo siempre que lo he necesitado. Espero no haber sido muy “pesada”.

También quiero agradecerles a mis padres por haberme apoyado siempre y estar tan seguros de mí. Es agradable tener a gente que cree en ti. Gracias por todo.

En esta universidad, aparte de estudiar y hacer exámenes, también he conocido a algunas de mis mejores amigas, con las que sufrí días enteros en la biblioteca haciendo trabajos y muchas colas del microondas en la cafetería. Gracias por haberme hecho más llevaderos estos años.

Resumen

En este Proyecto se pretende obtener una estimación de los parámetros característicos de una fisura de frente elíptico para diferentes disposiciones de cargas. Estas estimaciones se obtendrán mediante la utilización de Redes Neuronales Artificiales (RNA).

Para poder llevar esto a cabo, primero se realizarán los diferentes modelos para las distintas disposiciones de cargas (tracción, flexión y la combinación de ambas) en un programa de elementos finitos (Abaqus 6.7) y de este modo poder calcular el tamaño de la fisura para diferentes valores de los parámetros característicos de la fisura. Los parámetros que se utilizarán para el estudio serán la longitud característica de la fisura (α), el factor de forma (β) y el área de la fisura en tanto por uno del área total del eje. Una vez obtenidos estos valores se desarrollarán varias redes neuronales artificiales, para poder calcular el problema inverso, es decir, a partir del tamaño de la fisura poder estimar, mediante la red, los valores de los parámetros característicos mencionados anteriormente.

A la vista de los resultados obtenidos, se puede decir que el mayor grado de precisión en la estimación se da tanto en el área de la fisura como en la longitud característica, mientras que el factor de forma no puede ser estimado mediante redes neuronales artificiales. Esto es debido a que el factor de forma, al tratarse de una fisura de frente elíptico, ofrece multitud de posibles curvaturas y eso hace muy complicada su estimación.

Palabras clave: Fisuras de frente elíptico, Ejes, Redes Neuronales Artificiales.

Abstract

In this Proyect it is tried to get an estimation of some elliptical front crack's characteristic parameters for different loads provisions. These estimations will be obtained by using Artificial Neural Networks (ANN).

In order to accomplish this, different models will be implemented for different loads dispositions (traction, flexion and both of them together), using the Method of Finite Elements (Abaqus) and this way be able to calculate the crack's opening for different values of its characteristic parameters. The parameters that will be used for this study are the characteristic length (α), the form factor (β) and the area of the crack. Once these values are calculated, various Artificial Neural Networks (ANN) will estimate the parameters beginning with the values of the crack's opening.

At the view of the results, it can be said that the characteristic length and the area of the crack are parameters that can be estimated by the Artificial Neural Networks but it isn't the same for the other parameter. In the case of the form factor it isn´t possible to estimate the values due to the elliptical form of the front of the crack.

Keywords: Elliptical front cracks, Shafts, Artificial Neural Networks.

Índice general

Capítulo 1. Introducción y objetivos…….1

1.1 Introducción…….....…….....1
1.2 Objetivos…….....…….....…….3
1.3 Estructura del documento……...4

Capítulo 2. Planteamiento teórico del problema……...6

2.1 Comportamiento de ejes fisurados……..6
2.2 Geometría del frente de la fisura…….8

Capítulo 3. Método de elementos finitos (MEF)…….10

3.1 Breve introducción al MEF…...……...10
3.2 Introducción a ABAQUS…….....…….....12

Capítulo 4. Redes neuronales artificiales (RNA)……16

4.1 Introducción a las redes neuronales artificiales (RNA)……16
4.1.1 Tipos de redes neuronales según su arquitectura…….....18
4.1.2 Tipos de redes neuronales según su aprendizaje……19
4.1.3 Redes perceptron multicapa……19
4.1.4 Algoritmo de retropropagación……...21

4.1.4.1 Razón de aprendizaje……..23
4.2 Introducción al *toolbox* de redes neuronales de Matlab……24

Capítulo 5. Metodología……..26

5.1 Diseño del modelo numérico……..……..26
5.1.1 Tracción…….....……..34
5.1.2 Flexión…….....…….....…35
5.1.3 Flexo-tracción…….....…….36
5.1.4 Cálculo del área de la fisura……38
5.2 Programación de las redes neuronales artificiales……...…39
5.2.1 Recopilación de datos……40
5.2.2 Adecuación de la estructura de la RNA……..41
5.2.3 Obtención de la red entrenada……42
5.2.3.1 Tracción……..43
5.2.3.2 Flexión……46
5.2.3.3 Flexo-tracción……49

Capítulo 6. Resultados y discusión……53

6.1 Resultados……53
6.1.1 Tracción…….....……..55
6.1.2 Flexión…….....…….....……57
6.1.3 Flexo-tracción…….....……59
6.2 Discusión de los resultados……...…...62

Capítulo 7. Conclusiones y trabajos futuros……....65

7.1 Conclusiones…….....……65
7.2 Trabajos futuros……66

Bibliografía……..68

Anexo…….....…….....……...70

Índice de figuras

Figura 2.1. Eje fisurado sometido a esfuerzos de tracción (N) y flexión (M)…….....6
Figura 2.2. Geometría de una fisura de frente elíptico [1]……..…….....9
Figura 2.3. Área de la fisura sombreada sobre el área total de la sección del eje [9].....9
Figura 4.1. Propagación y ponderación de señales en una red neuronal……..…...17
Figura 4.2. Agrupaciones de neuronas……..…...18
Figura 4.3a. Función sigmoidea o logística……...…….....……...20
Figura 4.3b. Función tangente hiperbólica……..…….....……20
Figura 4.3c. Función escalón……...…….....…….20
Figura 4.4. Esquema de aplicación del sesgo en una red perceptron multicapa…....….21
Figura 4.5. Esquema de funcionamiento del algoritmo de retropropagación…...…...…22
Figura 5.1. Geometría del eje……...…….....…….27
Figura 5.2a. Particiones para β = 0……...……28
Figura 5.2b. Particiones para β = 1……...……28
Figura 5.3. Particiones perpendiculares al eje……..…28
Figura 5.4. Eje biapoyado……...……..…29
Figura 5.5. Eje con un extremo empotrado……...……30
Figura 5.6. Carga vertical aplica en el eje……...……..31
Figura 5.7. Carga de tracción aplicada en el eje……...……....31
Figura 5.8. Vista del eje mallado…….……...……..32
Figura 5.9. Desplazamiento vertical medido para los casos en los que aparece esfuerzo de flexión…….....……...……....33
Figura 5.10. Apertura longitudinal de la fisura medida en los casos donde aparece esfuerzo de tracción…….....…….....…….....……33
Figura 5.11. Ejemplo de representación de datos reales y estimados por la red….....……....…42

Figura 5.12. Valores de α reales y estimados por la red (entrenamiento tracción)…..….....43
Figura 5.13. Valores de β reales y estimados por la red (entrenamiento tracción)…….....44
Figura 5.14. Valores del área de la fisura reales y estimados por la red (entrenamiento tracción)…….....…….....…….45
Figura 5.15. Valores de α reales y estimados por la red (entrenamiento flexión)…….....……46
Figura 5.16. Valores de β reales y estimados por la red (entrenamiento flexión)…….....……47
Figura 5.17. Valores del área de la fisura reales y estimados por la red (entrenamiento flexión)…….....…….....……..…….....48
Figura 5.18. Valores de α reales y estimados por la red (entrenamiento flexo-tracción)….....49
Figura 5.19. Valores de β reales y estimados por la red (entrenamiento flexo-tracción)….....50
Figura 5.20. Valores del área de la fisura reales y estimados por la red (entrenamiento flexo-tracción)……..…….....…….....…….51
Figura 6.1. Representación datos reales y estimados en el test de validación de α, β y el área de la fisura (tracción)……..55
Figura 6.2. Representación del error cometido en el test de validación de α, β y el área de la fisura (tracción)…….....……...56
Figura 6.3. Representación datos reales y estimados en el test de validación de α, β y el área de la fisura (flexión)…….....…….....……...57
Figura 6.4. Representación del error cometido en el test de validación de α, β y el área de la fisura (flexión)…….....……..…..58
Figura 6.5. Representación datos reales y estimados en el test de validación de α, β y el área de la fisura (flexo-tracción)……....59
Figura 6.6. Representación del error cometido en el test de validación de α, β y el área de la fisura (flexo-tracción)…….....60

Índice de tablas

Tabla 5.1. Propiedades del material…….....…….27

Tabla 5.2a. Valores de la apertura longitudinal de la fisura adimensionalizados en tracción (con β desde 0 hasta 0.4)….…….34

Tabla 5.2b. Valores de la apertura longitudinal de la fisura adimensionalizados en tracción (con β desde 0.5 hasta 1)…….....34

Tabla 5.3a. Valores del desplazamiento vertical máximo del eje adimensionalizados en flexión (con β desde 0 hasta 0.4)…….……..35

Tabla 5.3b. Valores del desplazamiento vertical máximo del eje adimensionalizados en flexión (con β desde 0.5 hasta 1)……...35

Tabla 5.4a. Valores de la apertura longitudinal de la fisura adimensionalizados en flexo-tracción (con β desde 0 hasta 0.4)…….…...……...36

Tabla 5.4b. Valores de la apertura longitudinal de la fisura adimensionalizados en flexo-tracción (con β desde 0.5 hasta 1)….……36

Tabla 5.5a. Valores del desplazamiento vertical máximo del eje adimensionalizados en flexo-tracción (con β desde 0 hasta 0.4)…….....37

Tabla 5.5b. Valores del desplazamiento vertical máximo del eje adimensionalizados en flexo-tracción (con β desde 0.5 hasta 1)…….....37

Tabla 5.6a.Valores del área de la fisura en tanto por uno del área total de la sección del eje (con β desde 0 hasta 0.4)……...…….....38

Tabla 5.6b. Valores del área de la fisura en tanto por uno del área total de la sección del eje (con β desde 0.5 hasta 1)……..39

Tabla 5.7. Valor del error cuadrático medio de la estimación cometido por la red durante el entrenamiento……...…….....…….52

Tabla 6.1.ECM de la estimación de α, β y el área de la fisura para el test de validación.....61

Tabla 6.2. R^2 para los valores reales y estimados de α, β y el área de la fisura utilizados en el test de validación…….....…….....……...61

Capítulo 1

Introducción y objetivos

1.1 Introducción

En el ámbito de la ingeniería es muy común encontrarse con elementos mecánicos que contienen defectos o fisuras. Estos defectos suelen aparecer debido a las condiciones en las que trabaja el componente y a los esfuerzos a los que está sometido, acortando su vida de servicio.

Para evitar el elevado coste de reemplazo ante un desperfecto o el daño que puede ocasionar la rotura súbita de estos componentes como consecuencia de la presencia de fisuras, numerosos investigadores trabajan en desarrollar métodos de detección e identificación de fisuras o defectos en elementos mecánicos, que no sean destructivos, salvando la integridad física del elemento.

En el caso de ejes giratorios, como consecuencia de la fatiga de los esfuerzos cíclicos a los que están sometidos, aparecen defectos tipo fisura en planos perpendiculares a la dirección de éstos, lo que provoca el aumento de la flexibilidad del componente produciéndose cambios en el comportamiento estático y dinámico, que pueden traducirse en incrementos de desplazamientos transversales y disminución de las frecuencias de vibración, entre otros. Con el fin de detectar su presencia e identificar tanto su posición

como su tamaño, se han desarrollado diferentes trabajos, en los que se intenta relacionar el comportamiento mecánico de los ejes fisurados y la presencia de las fisuras.

El análisis del comportamiento de ejes fisurados sometidos a diferentes tipos de esfuerzos es un tema de gran importancia en los distintos campos de la ingeniería por el problema que comporta la ruptura catastrófica de los mismos, tanto en el caso de ejes sometidos a cargas estáticas, como en el de ejes giratorios en el que el problema se complica debido a la apertura y cierre de la fisura.

En la mayoría de los trabajos en los que se estudia el comportamiento de ejes fisurados, se supone que la fisura tiene frente recto, pero la experiencia demuestra que las fisuras de fatiga que aparecen en ejes suelen presentar un frente aproximadamente elíptico.

Lo más común en la determinación de la forma y el tamaño de fisuras es resolver el problema de forma directa, es decir, a partir de unos valores de los parámetros característicos del frente de la fisura, obtener la apertura longitudinal de la misma o el valor del desplazamiento vertical del eje en ese punto. Pero también es posible resolver el problema inverso, a partir de los valores de los desplazamientos producidos en el eje, poder obtener los valores de los parámetros característicos de la fisura.

Actualmente en defectología, las Redes Neuronales Artificiales (RNA) son utilizadas para la determinación de defectos en componentes mecánicos. Las RNA son estructuras matemáticas muy flexibles, capaces de identificar relaciones no lineales complejas entre unos valores de entrada y salida establecidos.

Por otra parte, es de gran interés resaltar el considerable aumento de programas destinados al diseño mediante computador, en concreto al método de los elementos finitos, que permiten realizar numerosos y difíciles cálculos, además del reducido coste que supone frente a la construcción de prototipos reales.

1.2 Objetivos

El objetivo del presente proyecto es poder estimar las características de una fisura de frente elíptico en un eje de sección circular, resolviendo el problema inverso mediante la utilización de redes neuronales artificiales (RNA).Esto quiere decir que, a partir de los valores de la apertura longitudinal de la fisura y del desplazamiento vertical sufrido en el eje, aplicando una RNA, se obtendrán los valores de diferentes parámetros característicos del frente de la fisura. Los parámetros seleccionados para realizar este estudio son:

- **Parámetros de entrada**
  - La apertura longitudinal de la fisura, en los casos en los que aparezcan esfuerzos de tracción.
  - La flecha vertical máxima del eje, en los casos en que aparezcan esfuerzos de flexión.
- **Parámetros de salida**
  - Longitud característica (α).
  - Factor de forma (β).
  - Área de la fisura en tanto por uno del área total de la sección del eje.

Estos cálculos se llevarán a cabo para un eje sometido a un tipo de esfuerzo distinto en cada caso.

- Tracción
- Flexión
- Flexo-tracción

Para llevar esto a cabo han de seguirse los siguientes puntos:

- Desarrollar los modelos numéricos para las diferentes disposiciones de cargas y condiciones geométricas, mediante un programa de análisis de elementos finitos.

- Una vez realizados los modelos numéricos, obtener los valores del desplazamiento vertical producido en el eje y de la apertura longitudinal de la fisura, para una serie de valores dados de varios parámetros característicos del frente de la misma. Estos parámetros serán los antes mencionados:

  - Longitud característica (α).
  - Factor de forma (β).
  - Área de la fisura en tanto por uno del área total de la sección del eje.

- Desarrollar las diferentes redes neuronales artificiales para resolver el problema inverso. Se establecerán como entradas los valores de la amplitud longitudinal de la fisura y del desplazamiento vertical del eje y como valores de salida los valores de los parámetros anteriores.

- Comparar los resultados obtenidos mediante análisis de elementos finitos con los obtenidos mediante las redes neuronales. Calcular el error cometido en las estimaciones de las redes.

- Discutir los resultados obtenidos.

1.3 Estructura del documento

En el capítulo 2 se muestra una breve introducción sobre los conceptos teóricos que son la base de los cálculos realizados en este proyecto y se definen los parámetros característicos de una fisura de frente elíptico, necesarios para poder realizar este estudio.

En el capítulo 3 se muestra brevemente en qué consiste el método de elementos finitos y de forma general, los pasos que han de seguirse para definir un modelo numérico en ABAQUS (el software utilizado para aplicar este método en el presente proyecto).

El capítulo 4 comienza con los principios teóricos de las redes neuronales artificiales (RNA) y los diferentes tipos de las mismas, atendiendo a sus diferentes características.

En el capítulo 5 se describe la metodología seguida para realizar los modelos numéricos en ABAQUS y su posterior análisis por elementos finitos. Seguidamente se expone el desarrollo de las redes neuronales para cada uno de los diferentes casos sometidos a estudio: tracción, flexión y la combinación de ambas.

En el capítulo 6 se recogen los resultados obtenidos mediante la aplicación de las redes neuronales y los errores cometidos por las mismas durante la estimación. Tras mostrar los resultados se puede encontrar la discusión de los mismos.

Las conclusiones finales obtenidas de los resultados y los proyectos futuros que podrían ser interesantes forman el capítulo 7.

Finalmente, se encuentran la bibliografía y el anexo. Este último recoge los códigos de programación de cada una de las redes neuronales artificiales utilizadas.

Capítulo 2

Planteamiento teórico del problema

2.1 Comportamiento de ejes fisurados

La aparición de fisuras en un elemento mecánico provoca un aumento de la flexibilidad local del elemento, causando cambios en su comportamiento, tanto estático como dinámico, que se traducen en el incremento de los desplazamientos, disminución de las frecuencias de vibración y modificación de las órbitas que describen sus secciones, entre otros.

Figura 2.1: *eje de sección circular fisurado, sometido a esfuerzos de tracción (N) y flexión (M)*

La existencia de una fisura en un elemento mecánico sometido a un esfuerzo de tracción *N* y a un momento flector *M* (figura 2.1), se puede modelizar aceptando discontinuidades, tanto en los desplazamientos longitudinales (Δu), como en los giros ($\Delta\theta$), en la sección fisurada, relacionados con los esfuerzos transmitidos por:

$$\Delta u = \lambda_{nn} N + \lambda_{nm} M \qquad (2.1)$$

$$\Delta\theta = \lambda_{mm} M + \lambda_{mn} N \qquad (2.2)$$

donde λ_{ij} son los coeficientes de flexibilidad y

$$\Delta u = u_2 - u_1 \qquad (2.3)$$

$$\Delta\theta = \theta_2 - \theta_1 \qquad (2.4)$$

Correspondiéndose u_1 y θ_1 al desplazamiento y el giro de la sección a la izquierda de la fisura y u_2 y θ_2 al desplazamiento y el giro de la sección a la derecha de la fisura.

Los coeficientes de flexibilidad del elemento fisurado se obtienen a partir de la relación entre la tasa de liberación de energía *G* y el factor de intensidad de tensiones K_I, que tiene la expresión:

$$G_N = \frac{1-\upsilon^2}{E} K_{I,N}^2 = \frac{N^2}{2}\frac{d\lambda_{nn}}{dA} \qquad (2.5)$$

$$G_M = \frac{1-\upsilon^2}{E} K_{I,M}^2 = \frac{M^2}{2}\frac{d\lambda_{mm}}{dA} \qquad (2.6)$$

con *E* igual al módulo de Young y υ igual al coeficiente de Poisson del material y dA igual al diferencial de área de la fisura.

Integrando estas expresiones se obtienen los coeficientes de flexibilidad para flexión, tracción y flexo-tracción, respectivamente:

$$\lambda_{mm} = \frac{2\left(1-\nu^2\right)}{E}\int_A \left(\frac{K_{I,M}}{M}\right)^2 dA \qquad (2.7)$$

$$\lambda_{nn} = \frac{2\left(1-\nu^2\right)}{E}\int_A \left(\frac{K_{I,N}}{N}\right)^2 dA \qquad (2.8)$$

$$\lambda_{mn} = \frac{2\left(1-\nu^2\right)}{E}\int_A \left(\frac{K_{I,N}}{N}\right)\left(\frac{K_{I,M}}{M}\right) dA \qquad (2.9)$$

L. Rubio y B. Muñoz en su artículo [1] desarrollan con detalle la obtención de expresiones de la flexibilidad de un eje con fisura elíptica en función del tamaño y la forma de la misma, a partir de ajustes polinómicos del factor de intensidad de tensiones.

2.2 Geometría del frente de la fisura

Es común suponer que las fisuras que se forman debido a fatiga en ejes de sección circular son de frente recto, para así simplificar el problema. Esto da resultados aceptables, pero experimentalmente se puede observar que las fisuras por fatiga que aparecen en ejes rotatorios tienen frente elíptico.

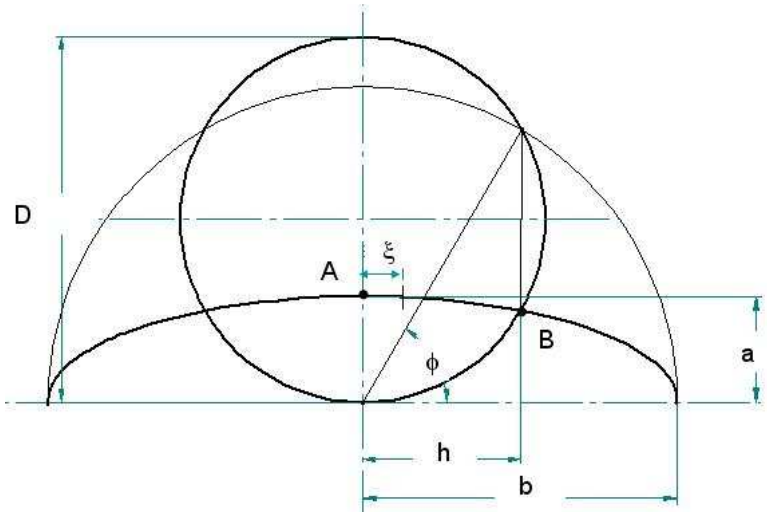


Figura 2.2: *geometría de una fisura de frente elíptico* [1].

Los parámetros más representativos de una fisura de frente elíptico (figura 2.2) son:

- $\alpha = \frac{a}{D}$ $\rightarrow$ longitud característica de la fisura (profundidad de la fisura)
- $\beta = \frac{a}{b}$ $\rightarrow$ factor de forma de la fisura (β = 1 se trata de una fisura de frente semicircular y β = 0 de una de frente recto)
- $\gamma = \frac{\xi}{h}$ $\rightarrow$ posición relativa en el frente de la fisura
- Área de la fisura (figura 2.3)

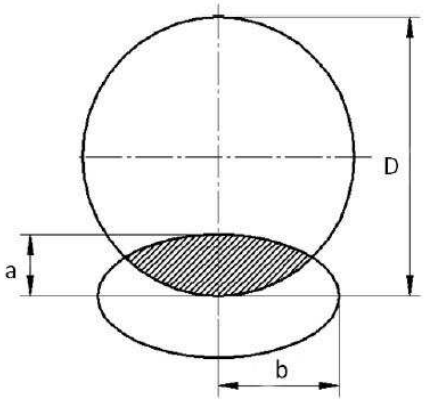


Figura 2.3: *área de la fisura sombreada sobre el área total de la sección del eje* [8]

Capítulo 3

Método de elementos finitos (MEF)

3.1 Breve introducción al MEF

El Método de los Elementos Finitos (MEF) es una de las herramientas más potentes usadas en la actualidad para la resolución numérica de un gran número de problemas de ingeniería. Este método es aplicable en una gran variedad de problemas, como pueden ser análisis estructurales, comportamiento mecánico de automóviles, problemas de transferencia de calor, electromagnéticos, etc.

Mediante el Método de Elementos Finitos se realiza una aproximación para obtener la solución de problemas continuos, basada en transformar un cuerpo de naturaleza continua en un modelo discreto aproximado. A esta transformación se le denomina discretización del modelo.

El continuo se divide en un número finito de partes denominados elementos. Las propiedades del material y sus ecuaciones constitutivas son consideradas sobre dichos elementos, los cuales poseen unos puntos característicos denominados nodos. Estos nodos son los puntos de unión de cada elemento con sus adyacentes.

El comportamiento en el interior de cada elemento queda definido a partir del comportamiento de los nodos mediante las "funciones de interpolación" o "funciones de forma". Estas funciones definen de manera única el campo de desplazamientos dentro de cada elemento finito, expresado en términos de los desplazamientos nodales de dicho elemento. Es por tanto, una aproximación de los valores de una función a partir del conocimiento de un número determinado y finito de puntos.

Si bien las verdaderas funciones de forma son desconocidas, se puede sentar la hipótesis de que su expresión aproximada puede ser obtenida en forma polinómica.

La relación entre estos elementos, considerando debidamente las condiciones de contorno (cargas y restricciones), da lugar a un sistema de ecuaciones cuya solución va a permitir obtener resultados mediante los cuales se va a conocer el comportamiento aproximado del continuo.

El Método de Elementos Finitos puede dividirse en tres etapas [6]:

- Preproceso. Preparación del modelo para el cálculo. En esta etapa se realizan las operaciones de:

  - Dibujo de la geometría del modelo.
  - Selección de las propiedades de los materiales.
  - Aplicación de cargas exteriores y condiciones de contorno.
  - Discretización del modelo en elementos finitos.

- Resolución. Etapa en la que se realizan todos los cálculos y se generan las soluciones. En ella se realizan las operaciones de:

  - Selección del tipo de cálculo a realizar.
  - Configuración de los parámetros de cálculo, intervalos de tiempo y número de iteraciones.
  - Transferencia de las cargas al modelo, generación de funciones de forma, ensamblaje de la matriz de rigidez, resolución de sistemas de ecuaciones y obtención de la solución.

- Postproceso. En esta etapa se realizará la representación gráfica de los resultados, así como la obtención de resultados indirectos operando las soluciones del modelo.

3.2 Introducción a ABAQUS

ABAQUS es un código de análisis por el Método de los Elementos Finitos de propósito general, orientado a la resolución de problemas no lineales. Fue desarrollado hace más de 20 años por la empresa Hibbit, Karlsson & Sorensen, Inc. (HKS), y en la actualidad se utiliza para resolver grandes y complejos problemas de ingeniería. ABAQUS puede ser utilizado para resolver problemas de resistencia de materiales, mecánica de fractura, ingeniería forense, procesos de conformado de metales, transferencia de calor, etc.

ABAQUS está estructurado en tres grandes bloques, en correspondencia con las tres etapas en que se divide un problema para ser analizado por el Método de Elementos Finitos (preproceso, resolución y postproceso).

ABAQUS se encuentra dividido en cuatro módulos [6]:

- ABAQUS/Standard, para resolución de problemas de propósito general. Incluye todas las posibilidades de análisis excepto el análisis dinámico.

- ABAQUS/Explicit, para resolución de problemas de tipo dinámico. Es poderoso por su eficiencia computacional en grandes modelos, y también altamente efectivo para aplicaciones cuasi-estáticas.

- ABAQUS/CAE, módulo interactivo para la creación de modelos de elementos finitos. A través de este módulo se puede implementar un modelo de forma sencilla y rápida.

- ABAQUS/Viewer, módulo de visualización de soluciones. Muestra los resultados obtenidos una vez resuelto el problema.

Para resolver un problema mediante ABAQUS 6.7 se deben introducir los datos de entrada que necesita el programa. Esta etapa se corresponde con la etapa de preproceso del Método de los Elementos Finitos.

Los datos de entrada se introducen en el programa a través de un archivo de texto (*input file*) que contiene toda la información necesaria para realizar la simulación. Este archivo de texto, también denominado "archivo de entrada", se puede crear de forma interactiva utilizando ABAQUS/CAE, que genera internamente el archivo de texto, o mediante un editor de texto, donde el archivo es escrito directamente por el usuario.

Los datos que se introducen en el archivo de entrada son de dos tipos, *model data* y *history data*. Los *model data* que se utilizan para definir el modelo de elementos finitos son:

- *Geometría*. La geometría es lo primero que se introduce y debe representar lo más fielmente posible el cuerpo real que se va a estudiar. ABAQUS permite realizar modelos de geometría muy compleja, gracias al módulo ABAQUS/CAE. La geometría, así como el mallado de un modelo, se define mediante los elementos y sus nodos.

- *Material*. El modelo que se va a implementar puede estar formando de varios materiales. Se deben definir las propiedades de los distintos materiales de los que está formado nuestro modelo, y se deben asociar a las partes de la geometría a las que correspondan.

- *Partes y ensamblaje*. La geometría del modelo se puede definir organizándola en partes, por lo tanto, hay que introducir en el archivo de entrada la relación que existe entre esas partes, además de la posición relativa de unas respecto de las otras. Esto se lleva a cabo a través de un procedimiento conocido como ensamblaje.

- *Condiciones iniciales*. Se deben especificar las condiciones iniciales del modelo. En ocasiones, es necesario especificar condiciones iniciales distintas de cero para tensiones, temperaturas, velocidades, etc.

- *Condiciones de contorno*. El modelo puede estar sometido a ciertas imposiciones por parte de su entorno que deben ser especificadas en el archivo de entrada. Se pueden imponer restricciones del movimiento, valores de desplazamientos y rotaciones o condiciones de simetría.

- *Interacciones*. En ocasiones el modelo que se desea estudiar puede estar formado por varios cuerpos que en un momento dado sufren una interacción o un contacto. ABAQUS permite también realizar modelos ante este tipo de situaciones.

- *Definiciones de amplitud*. Ciertos estados de carga y condiciones de contorno pueden estar definidas en función del tiempo, por lo tanto se deben introducir los parámetros de las curvas que definen esos estados de cargas y condiciones de contorno.

- *Propiedades del entorno*. Se pueden definir las características del entorno, como puede ser la humedad, temperatura, presión, etc.

- *Continuación de análisis.* Consiste en introducir resultados de análisis previos para continuar buscando resultados con el nuevo modelo.

Después de introducir el archivo de entrada en el programa, se procede a la resolución del modelo. Esta fase es interna y en ella ABAQUS no interacciona con el usuario. Una vez que ABAQUS ha resuelto el modelo, los resultados de la simulación se ven a través del módulo de visualización, el cual, lee el archivo de datos de salida (*output file*), y es capaz de crear animaciones de la simulación, gráfico, tablas de resultados, etc. La visualización de los resultados se corresponde con la etapa de postproceso del Método de Elementos Finitos.

Capítulo 4

Redes neuronales artificiales (RNA)

4.1 Introducción a las redes neuronales artificiales (RNA)

Las RNA son otra forma de emular una de las características propias de los humanos: la capacidad de memorizar y asociar hechos. Si analizamos aquellos problemas que no pueden expresarse a través de un algoritmo, nos daremos cuenta de que todos ellos tienen una característica en común: la experiencia. El hombre es capaz de resolver estas situaciones acudiendo a la experiencia acumulada. En definitiva, las redes neuronales no son más que un modelo artificial simplificado del cerebro humano, que es el ejemplo más perfecto que tenemos de sistema capaz de adquirir conocimiento a través de la experiencia.

El cerebro humano contiene alrededor de 12 billones de neuronas. Cada una de ellas tiene entre 5.600 y 60.000 conexiones dendríticas. Estas conexiones transportan los impulsos enviados desde unas neuronas a otras. Cada neurona tiene una salida denominada axón. El contacto de cada axón con una dendrita se realiza a través de la sinapsis. Tanto el axón como las dendritas transmiten la señal en una única dirección.

Las redes neuronales artificiales basan su funcionamiento en las redes neuronales reales, estando formadas por un conjunto de unidades de procesamiento conectadas entre sí. Por analogía con el cerebro humano, se denomina "neurona" a cada una de estas unidades de procesamiento. Cada "neurona" recibe muchas señales de entrada y envía una única señal de salida, al igual que las neuronas reales [4].

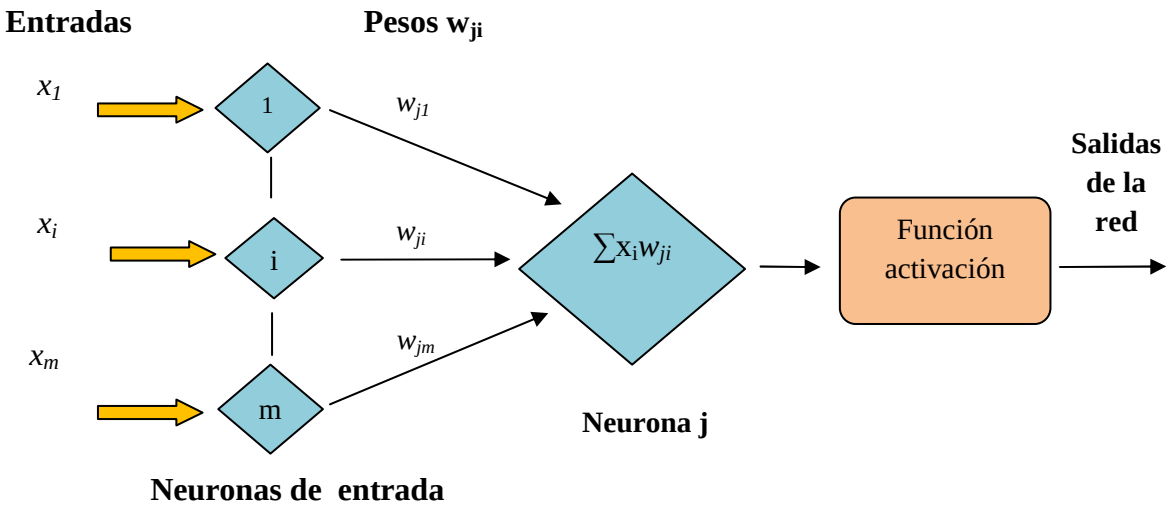


Figura 4.1: *propagación y ponderación de señales en una red neuronal*

Los pesos w_{ij} multiplican los valores de entrada de la red que reciben las neuronas, haciéndolos más o menos importantes. Las entradas, una vez ponderadas, se suman en la neurona *j*. Este valor final, entra en una función que activa el valor que pasará a la siguiente neurona (figura 4.1).

Existen diferentes tipos de redes neuronales atendiendo a la estructura de la red o al tipo de aprendizaje al que se las someta.

4.1.1 Tipos de redes neuronales según su arquitectura

En las RNA, las neuronas se agrupan en capas que se organizan para formar la estructura de la red (figura 4.2).

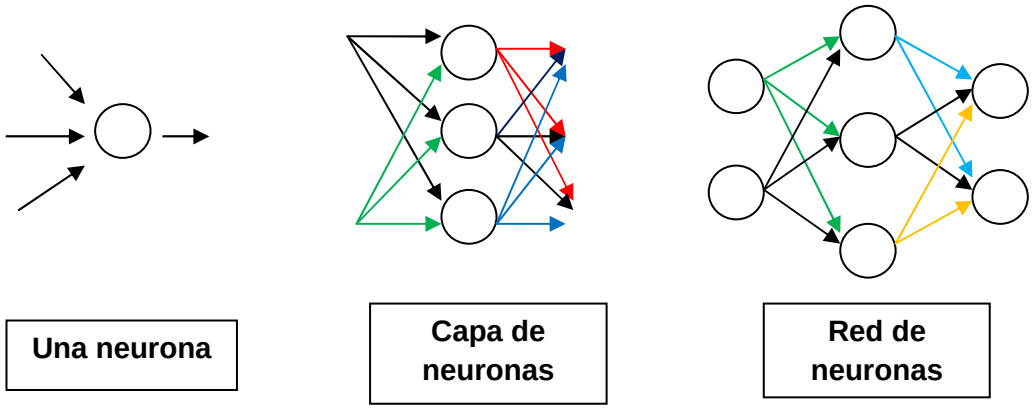


Figura 4.2: *agrupaciones de neuronas*

- **RNA monocapa:** como su propio nombre indica, las neuronas se agrupan formando una sola capa. Estas redes son ampliamente utilizadas en circuitos eléctricos, ya que de debido a su arquitectura, son adecuadas para ser implementadas mediante hardware, usando matrices de diodos que representan las conexiones neuronales.

- **RNA multicapa:** están formadas por varias capas de neuronas. Estas capas se pueden organizar según el orden en el que reciben la señal, desde la entrada hasta la salida (conexiones feedforward o hacia adelante) o en el orden inverso (conexiones feedback, hacia atrás o retroalimentadas).

4.1.2 Tipos de redes neuronales según su aprendizaje

La modelización con redes neuronales artificiales implica dos etapas fundamentales [2]: aprendizaje y generalización.

- **Aprendizaje:** consiste en el entrenamiento de la red con patrones, que usualmente son llamados patrones de muestra o entrenamiento. El proceso usual del algoritmo es que la red ejecute los patrones iterativamente, cambiando los pesos de las sinapsis, hasta que convergen a un conjunto de pesos óptimos que representan a los patrones lo suficientemente bien, entonces mostrará una respuesta satisfactoria. Hay tres tipos de aprendizaje:

    a) **Supervisado**: para cada patrón hay una respuesta deseada. La respuesta de la red se compara con la salida deseada. Se deben minimizar las diferencias entre las salidas de la red y los valores deseados.

    b) **No supervisado**: no se especifica la respuesta correcta. La red sigue una regla de autoorganización.

    c) **Reforzado**: se indica si la respuesta que da la red es correcta o no, pero sin especificar la respuesta que hay que obtener.

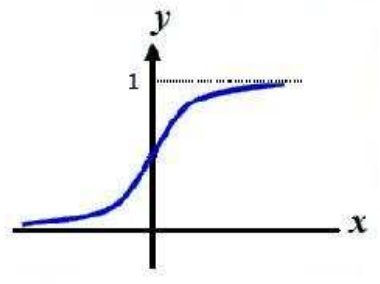
- **Generalización:** proceso de aplicar a la red ya aprendida, datos que no intervinieron en el aprendizaje.

4.1.3 Redes perceptron multicapa

En este tipo de redes la señal llega a la capa de entrada y pasa por una o varias capas de neuronas ocultas hasta llegar a la salida. En esas capas intermedias se aplica una función de activación, generalmente una función continua no lineal, que asigna un peso o ponderación a cada señal. Esta función calcula el estado de actividad de la neurona y transforma la entrada ponderada global en un valor de activación.

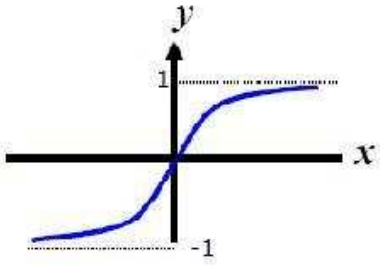
Usualmente los valores de activación varían entre (0, 1) ó (-1, 1), estando la neurona totalmente activa en (1) e inactiva en (-1 ó 0).

- Las funciones de activación más utilizadas son [2]:



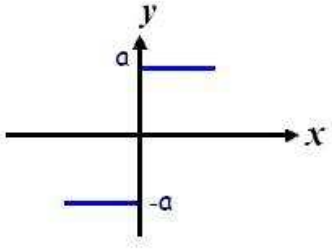
$$y_j = \frac{1}{1+e^{-\alpha \sum x_i w_{ji}}}$$

Figura 4.3a: *función sigmoidea o logística*



$$y_j = \frac{e^{\alpha \sum x_i w_{ji}} - e^{-\alpha \sum x_i w_{ji}}}{e^{\alpha \sum x_i w_{ji}} + e^{-\alpha \sum x_i w_{ji}}}$$

Figura 4.3b: *función tangente hiperbólica*



$$y_j = \begin{cases} a & \text{si} \sum x_i w_{ji} > 0 \\ a & \text{si} \sum x_i w_{ji} \leq 0 \end{cases}$$

Figura 4.3c: *función escalón*

En ocasiones puede ser interesante introducir un valor de *sesgo*, ya sea positivo o negativo, según convenga, que aumente o disminuya la activación. Para ello se introduce una entrada, x_0, que modifica el peso w_{j0}, y con ello su activación (figura 4.4).

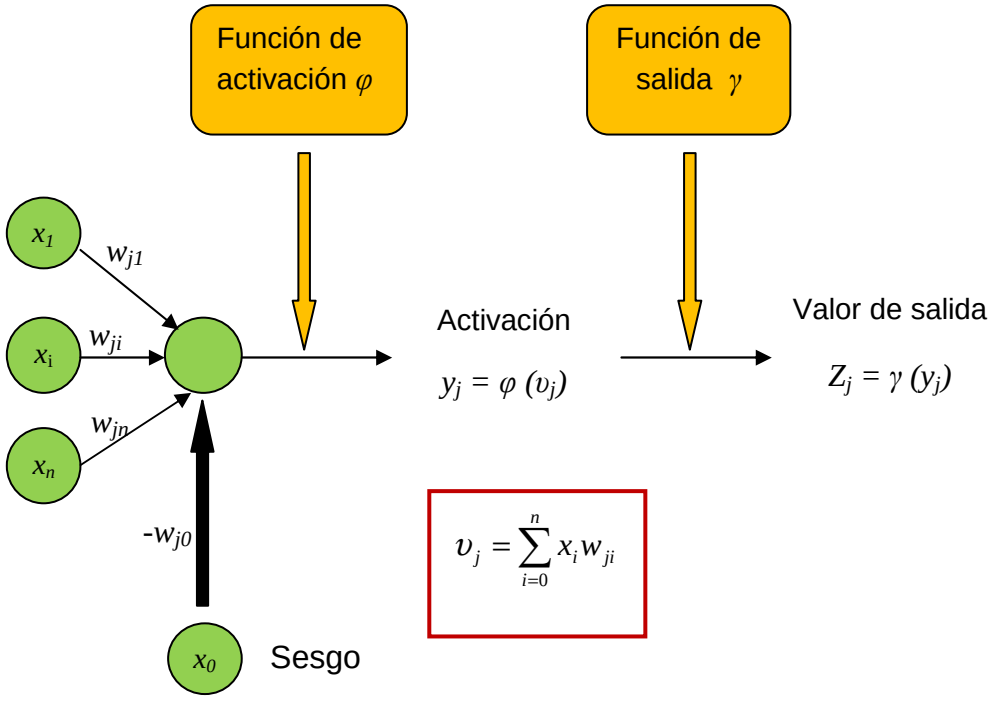


Figura 4.4: *esquema de aplicación del sesgo en una red perceptron multicapa*

4.1.4 Algoritmo de retropropagación

El algoritmo de retropropagación (backpropagation) está dividido en dos fases (figura 4.5):

1) **Fase hacia adelante**. Los pesos son fijos y se propagan de una capa a otra.

2) **Fase hacia atrás**. Los pesos se ajustan mediante una regla de corrección, que consiste en la propagación hacia atrás del error cometido en la salida de la red.

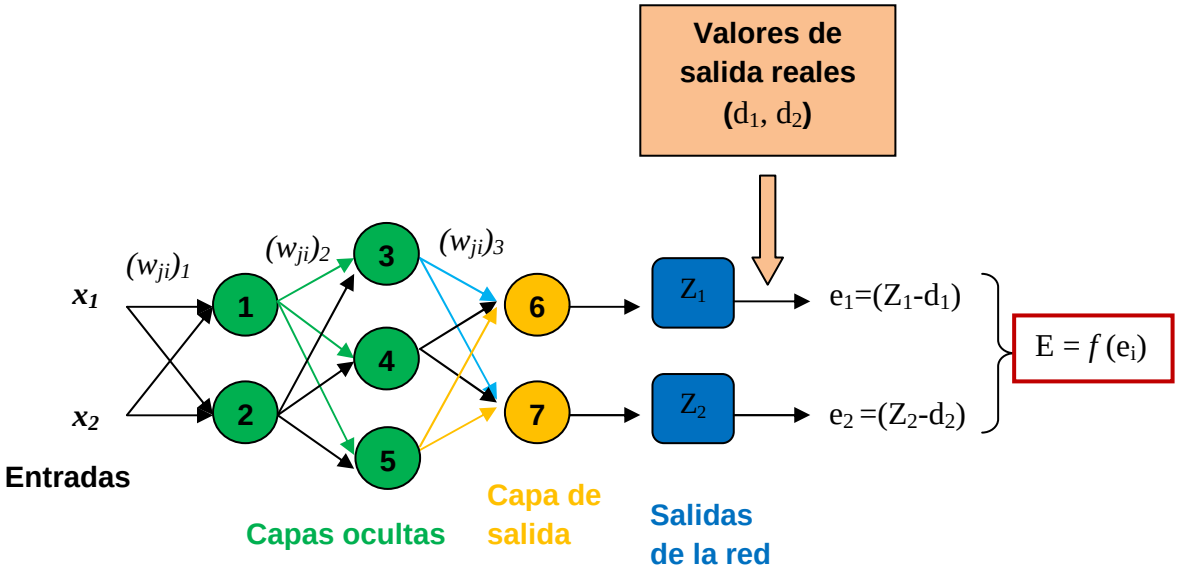


Figura 4.5: *esquema de funcionamiento del algoritmo de retropropagación*

En la figura 4.5 se puede observar el funcionamiento del algoritmo de retropropagación en una red neuronal que tiene dos capas ocultas.

La propagación del error se realiza en la dirección negativa del gradiente de la función error [2]:

$$E = \frac{1}{2}\sum\left(Z_j - d_j\right)^2 = \frac{1}{2}\sum\left(y_j - d_j\right)^2 \qquad (4.1)$$

En la iteración, la adaptación de los pesos se calcula como:

$$w_{ji(t)} = w_{ji(t-1)} + \Delta w_{ji(t)} \qquad (4.2)$$

En la que el cambio de los pesos en la neurona *j* requiere de la derivada de la función de activación de la neurona y del error cometido en ella.

$$\Delta w_{ji(t)} = -\eta \frac{\partial E_{(t)}}{\partial w_{ji}} = -\eta e_{j(t)} \varphi'_{j(t)}\, y_{ji} = \eta \delta_{j(t)} y_{ji} \qquad (4.3)$$

donde η es la razón de aprendizaje y δ es el gradiente local

Como el error en la neurona *j* de la capa de salida (superíndice *s*) es conocido, podemos obtener los pesos en la capa de salida:

$$w^{s}_{ki(t)} = -\eta \frac{\partial E(t)}{\partial w^{s}_{ki}} = -\eta e_{k(t)} \varphi'_{k(t)}\, y_{ki(t)} = -\eta \delta_{k(t)} y_{ki(t)} \qquad (4.4)$$

Donde y_{ki} se refiere a los valores de salida de las neuronas *i* de la capa anterior, que son las entradas de la neurona *k*.

Para calcular los errores de las neuronas de las capas ocultas (subíndice *j*) se debe hacer recursivamente mediante el algoritmo de propagación del error de las neuronas de la capa siguiente (subíndice *k*) conectadas directamente con ella.

$$\Delta w_{ji} = -\eta \varphi'_{j(t)} \sum_{k} \left(e_{k(t)} \varphi'_{k(t)}\, w_{kj(t)} \right) y_{ji(t)} = -\eta \varphi'_{j(t)} \sum_{k} \left(\delta_{k(t)} w_{kj(t)} \right) y_{ji(t)} \quad (4.5)$$

donde $\delta_{k(t)}$ es el gradiente local de las neuronas de la capa siguiente, conectadas a la neurona *j* de la capa oculta que se analiza.

4.1.4.1 Razón de aprendizaje (η)

La razón de aprendizaje es la que controla la magnitud del cambio que sufren los pesos en cada iteración. Con una **razón de aprendizaje pequeña**, se converge lentamente, en cambio, con una **razón de aprendizaje grande** se converge de manera más rápida, pero puede producirse inestabilidad.

Para poder evitar la inestabilidad al aumentar la razón de aprendizaje se introduce un término llamado *momento* (α) que tiene en cuenta el cambio del peso sináptico en la anterior iteración.

$$\Delta w_{ji(t)} = \alpha \Delta w_{ji(t-1)} + \eta \delta_{j(t)} y_{ji(t)} = \eta \sum_{n=0}^{t} \alpha_{(t-n)} \delta_{j(t)} y_{ji(t)} \qquad (4.6)$$

El aprendizaje se puede realizar de dos modos diferentes:

- Modo Batch: los pesos de la red se actualizan una vez que ya han pasado todos los patrones.

- Modo secuencial: los cambios de cada peso se calculan cada vez que pasa un patrón y cuando ya han pasado todos, se actualizan los pesos con el valor promedio de todos los cambios calculados.

4.2 Introducción al toolbox de redes neuronales de Matlab

Para poder aplicar la red neuronal a los datos obtenidos mediante el análisis de métodos finito realizado con Abaqus, se utilizará el programa de cálculo numérico Matlab.

Éste es un programa para realizar cálculos con vectores y matrices. Otra de sus ventajas es que puede trabajar con números escalares (reales o complejos), cadenas de caracteres y otras estructuras de información. Su atractivo principal es que puede realizar gran variedad de gráficos en dos y tres dimensiones y además posee un lenguaje de programación propio.

Para la realización de este proyecto se ha utilizado el toolbox de redes neuronales. Este toolbox está compuesto por una colección de funciones predefinidas en un ambiente

numérico de cómputo de Matlab. Estas funciones al ser llamadas por el usuario, simulan diferentes tipos de modelos neuronales. Los modelos de redes neuronales más populares son: perceptron, Adaline, backpropagation, de base radial, SOM, Elman, Hopfield y LVQ.

La realización de la red se hace en el editor de Matlab, empleando el código de programación específico del toolbox de redes neuronales. El usuario puede llamar a las distintas funciones para desarrollar la estructura de la red que más le convenga.

En este editor las funciones pertenecientes al módulo de redes neuronales aparecen en color morado, las funciones de cálculo de Matlab en color azul y el texto explicativo en color verde.

Capítulo 5

Metodología

5.1 Diseño del modelo numérico

Como ya se ha mencionado en el capítulo1, el objetivo de este proyecto es poder obtener los valores de diferentes parámetros característicos de una fisura de frente elíptico a partir de los valores de la apertura longitudinal de la misma y del desplazamiento vertical producido en el eje, en tres casos diferentes: sometiendo al eje a tracción, a flexión y a una combinación de ambas.

Para poder obtener los valores de los desplazamientos, tanto longitudinales como verticales del eje, se utilizará un software de análisis de elementos finitos, concretamente ABAQUS 6.7.

Los parámetros de la fisura que se utilizarán para el cálculo de los desplazamientos son: la longitud característica (α), el factor de forma (β) y el área de la fisura en tanto por uno del área total de la sección del eje. Se tomarán valores de β (de 0 a 1) con un intervalo de 0,1 y para cada valor de β se tomarán valores de α (de 0,05 a 0,5) en intervalos de 0,05. Los datos se encuentran recogidos en las tablas 5.2, 5.3, 5.4 y 5.5.

Después se calcularán los valores del área de la fisura para cada pareja de valores del factor de forma y de la longitud característica (ver tablas 5.6).

Los pasos seguidos para establecer los parámetros de cálculo en ABAQUS serán:

- *Definir la geometría:* se modelizará un eje cilíndrico de 20 mm de diámetro y 700 mm de longitud (figura 5.1).

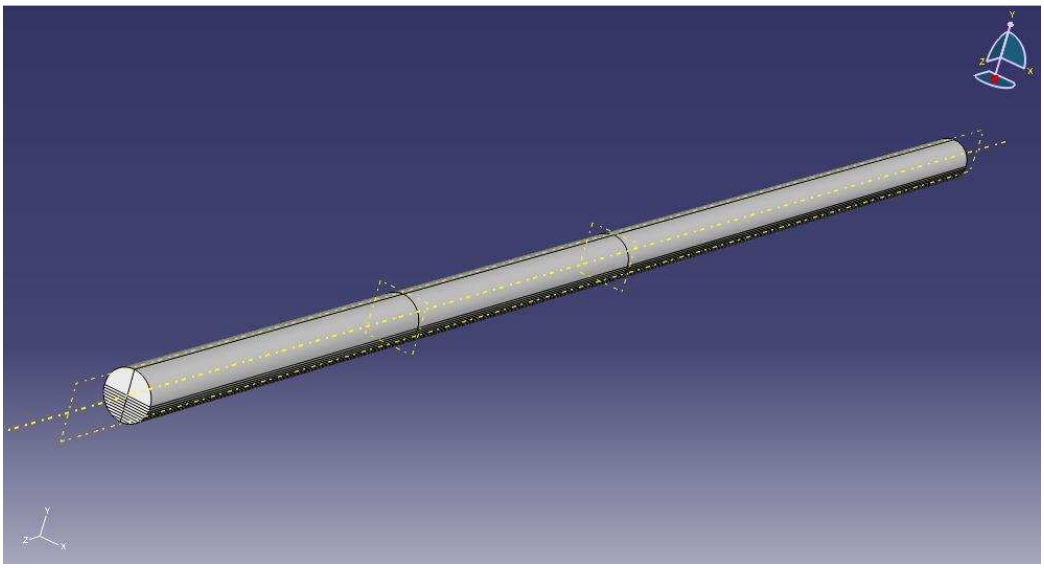


Figura 5.1: *geometría del eje*

- *Definir el material:* se seleccionará acero con las siguientes propiedades (tabla 5.1).

Módulo de Young (E)	**Coeficiente de Poisson (ν)**	**Densidad (ρ)**
210 GPa	0.3	7850 Kg/m^3

Tabla 5.1: *propiedades del material*

- *Secciones de interés:* para poder estudiar con mayor detalle la zona donde aparecerá la fisura y los parámetros característicos del frente de la misma, se realizarán particiones para luego obtener un mallado más fino. Se utilizarán dos tipos de particiones:

Paralelas al eje → creadas a lo largo de toda la mitad inferior del eje. Son de 5mm de alto, al igual que los intervalos de valores de α que se tomarán para el estudio. La forma de las particiones variará según el valor del factor de forma (β). Serán rectas para β = 0 e irán curvándose hasta ser circulares para β = 1 (ver figuras 5.2a y 5.2b).

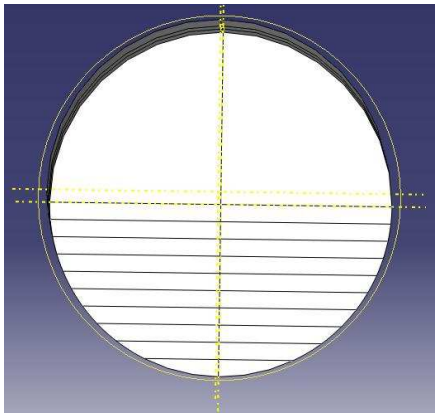


Figura 5.2a: *particiones para β=0*

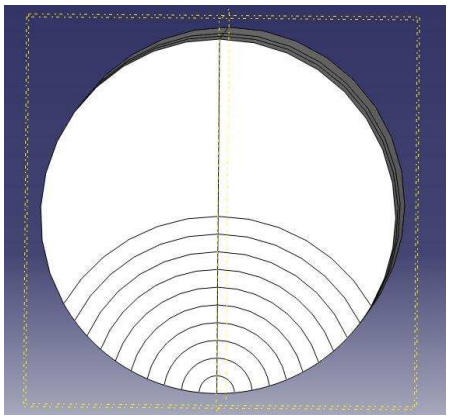


Figura 5.2b: *particiones para β=1*

Perpendiculares al eje → dos particiones creadas a lo largo del eje una en la mitad (L =350 mm) donde se aplicará la carga de flexión y otra en la sección donde aparecerá la fisura (L =180mm) como se muestra en la figura 5.3.

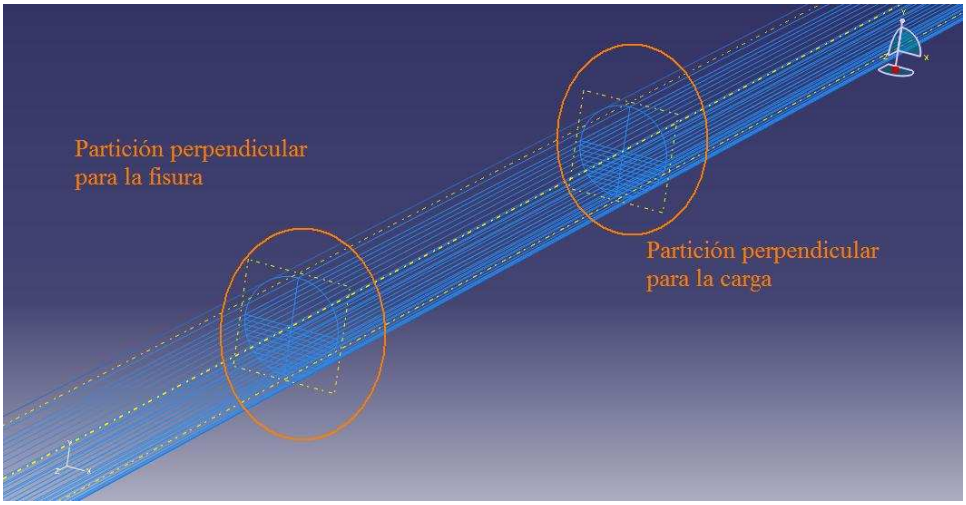


Figura 5.3: *particiones perpendiculares al eje*

- *Definir restricciones:* en este estudio se considerarán tanto esfuerzos de tracción como de flexión, por lo tanto, la combinación de restricciones será distinta dependiendo del caso que se estudie.

Flexión → para el caso de flexión se define un eje biapoyado en la mitad inferior de sus extremos.

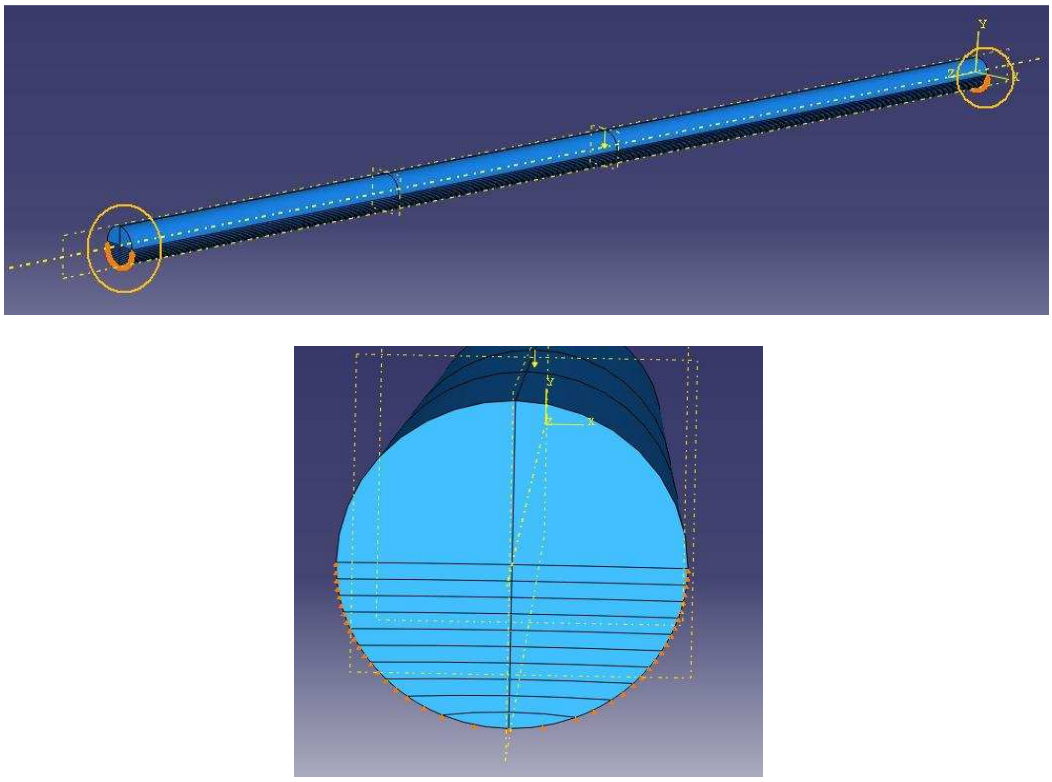


Figura 5.4: *eje biapoyado*

Tracción → para el caso de tracción se define un empotramiento en uno de los extremos del eje dejando libre el otro (figura 5.5).

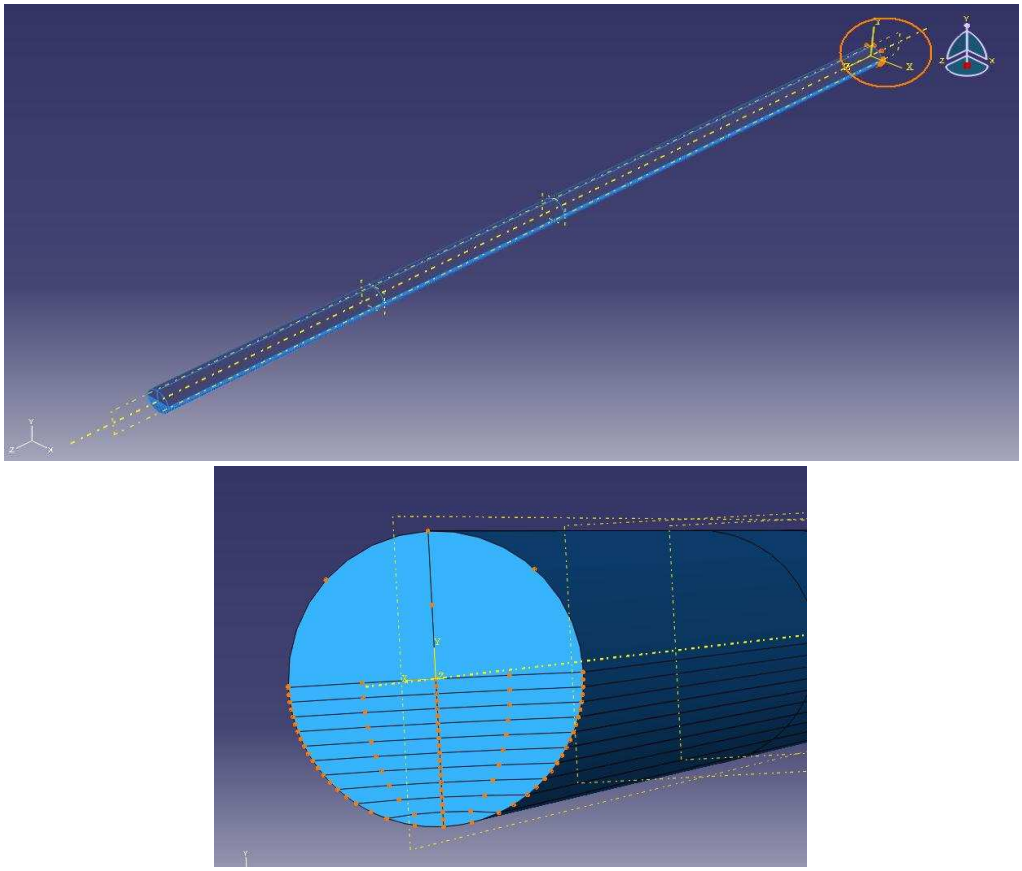


Figura 5.5: *eje con un extremo empotrado*

Flexo-tracción → para el caso de flexo-tracción se define la misma disposición que para el caso de tracción.

- *Aplicar las cargas:* al igual que en el punto anterior se tendrán diferentes combinaciones de cargas dependiendo del caso a considerar.

Flexión → se aplicará una carga puntual vertical hacia abajo de valor 150 N en la mitad del eje (figura 5.6).

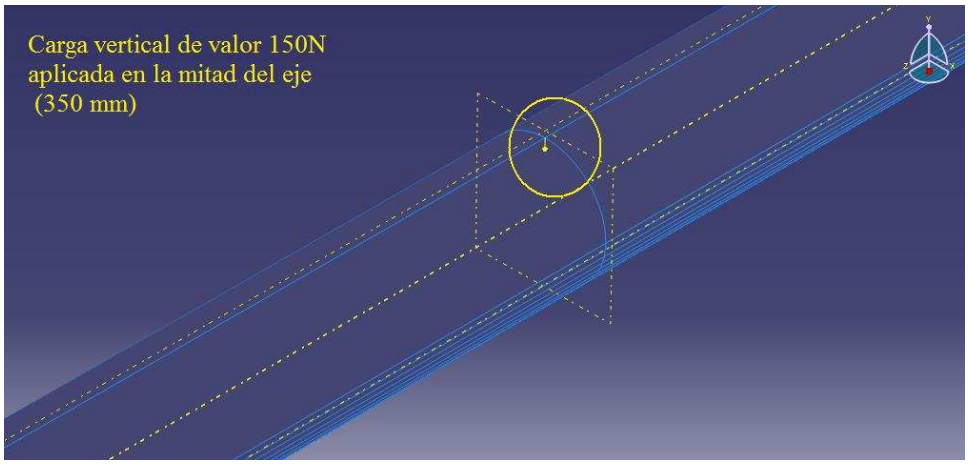


Figura 5.6: *carga vertical aplicada en el eje*

Tracción → se aplicará una carga horizontal uniformemente distribuida, de valor 10^7 N, en el extremo libre del eje (figura 5.7).

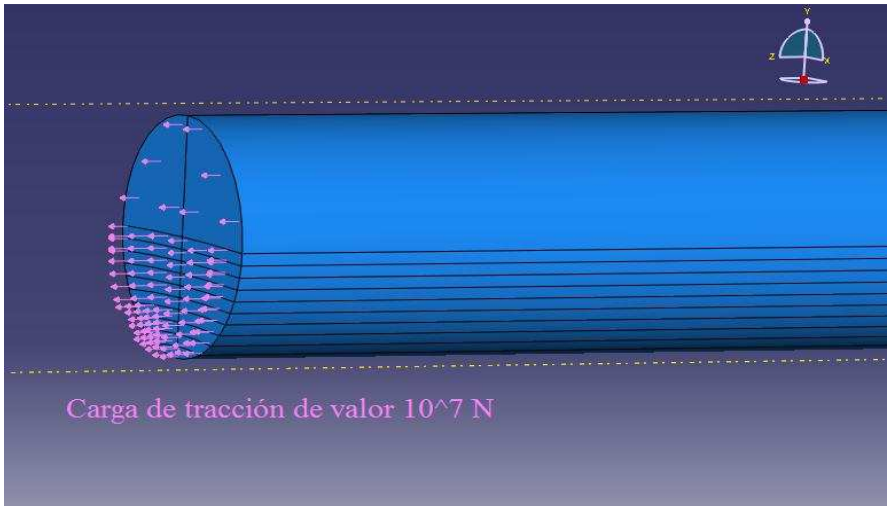


Figura 5.7: *carga de tracción aplicada en el eje*

Flexo-tracción → se aplicarán ambas cargas, tanto la vertical como la horizontal.

- *Mallado:* una vez que las cargas hayan sido aplicadas y las coacciones impuestas, se procederá al mallado de la estructura como se muestra en la figura 5.8.

Se han seleccionado elementos de mallado de tipo hexaédrico de aproximadamente 1,2 mm de profundidad medidos en un plano xy, con 8 nodos cada uno. Cada uno de los nodos tiene 3 grados de libertad (U_x, U_y y U_z). Las particiones realizadas en la mitad inferior del eje están realizadas de forma que se consigue un mallado más fino en la zona donde aparecerá la fisura. Estos elementos tienen un fondo de aproximadamente $240 \cdot 10^{-3}$ mm, también medido en un plano xy perpendicular al eje.

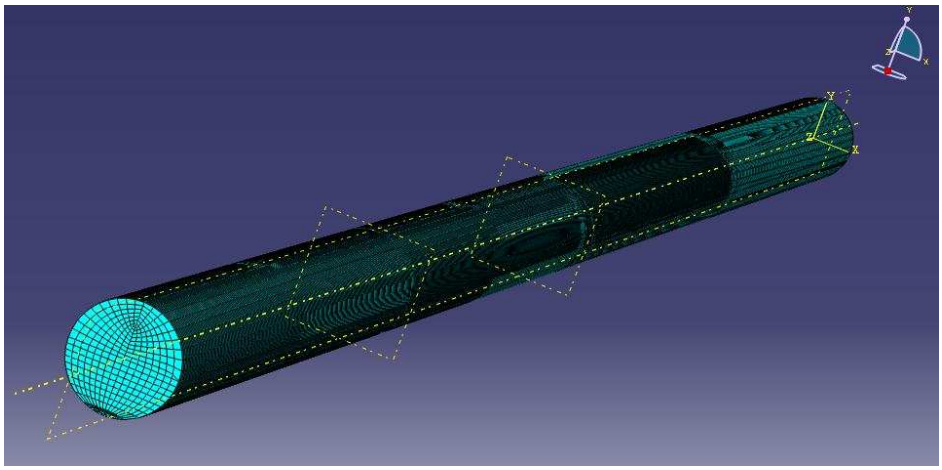


Figura 5.8: *vista del eje mallado*

- *Cálculos*: por último, tras haber establecido todas las condiciones y definido todos los parámetros necesarios, el programa procederá calcular la deformación del eje.

Una vez se tenga el modelo deformado calculado, se obtendrá el valor de la apertura longitudinal de la fisura para el caso de tracción y el desplazamiento vertical del eje en la sección fisurada para el caso de flexión. En el caso de flexo-tracción se tomarán ambas medidas, detalladas en las figuras 5.9 y 5.10.

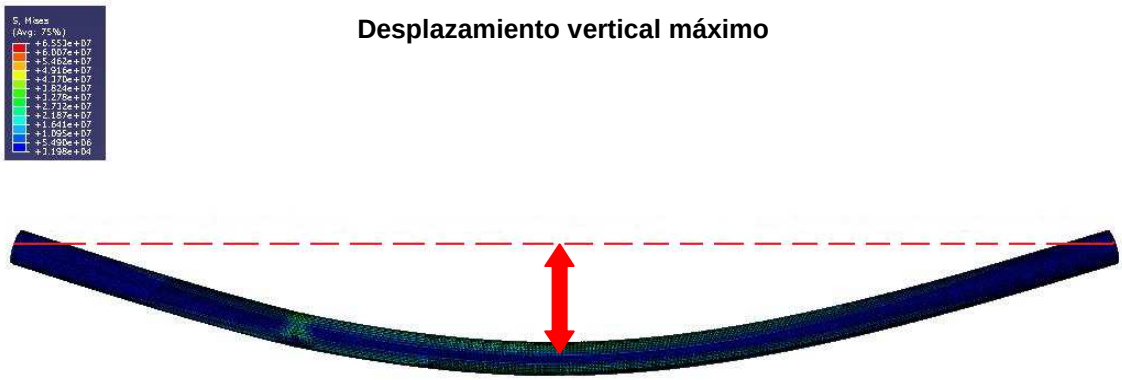


Figura 5.9: *desplazamiento vertical medido para los casos donde aparece esfuerzo de flexión*

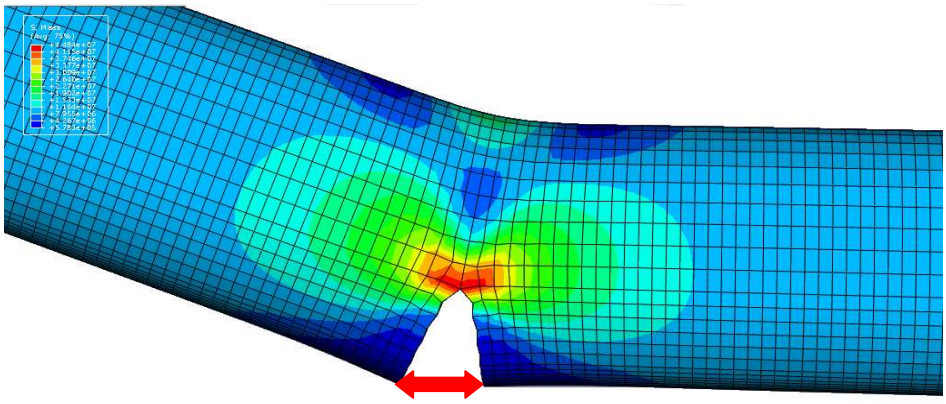


Figura 5.10: *apertura longitudinal de la fisura medida en los casos donde aparece esfuerzo de tracción*

A continuación se muestran las tablas que recogen los valores obtenidos de las medidas de los desplazamientos indicados en las figuras 5.9 y 5.10.

5.1.1 Tracción

Valores de la apertura longitudinal de la fisura, adimensionalizados con el desplazamiento longitudinal del eje sin fisura, cuando este está sometido a una esfuerzo de tracción.

α	β=0	β=0.1	β=0.2	β=0.3	β=0.4
0.05	0.008	0.007	0.007	0.006	0.006
0.1	0.015	0.015	0.014	0.013	0.014
0.15	0.023	0.022	0.022	0.021	0.020
0.2	0.032	0.033	0.031	0.032	0.029
0.25	0.046	0.044	0.045	0.042	0.042
0.3	0.061	0.063	0.060	0.061	0.056
0.35	0.087	0.084	0.085	0.080	0.079
0.4	0.119	0.120	0.116	0.116	0.108
0.45	0.172	0.167	0.167	0.159	0.156
0.5	0.243	0.245	0.238	0.235	0.222

Tabla 5.2a: *valores de la apertura longitudinal de la fisura adimesionalizados en tracción (con β desde 0 hasta 0.4)*

α	β =0.5	β =0.6	β =0.7	β =0.8	β =0.9	β =1
0.05	0.006	0.005	0.005	0.005	0.004	0.004
0.1	0.013	0.012	0.011	0.010	0.009	0.009
0.15	0.019	0.018	0.017	0.016	0.016	0.015
0.2	0.028	0.027	0.026	0.024	0.023	0.021
0.25	0.040	0.036	0.034	0.032	0.029	0.027
0.3	0.053	0.052	0.048	0.045	0.041	0.038
0.35	0.075	0.069	0.063	0.058	0.053	0.048
0.4	0.103	0.097	0.090	0.082	0.074	0.067
0.45	0.147	0.135	0.123	0.111	0.099	0.088
0.5	0.210	0.197	0.181	0.162	0.144	0.126

Tabla 5.2b: *valores de la apertura longitudinal de la fisura adimensionalizados en tracción (con β desde 0.5 hasta 1)*

5.1.2 Flexión

Valores del desplazamiento vertical máximo del eje, adimensionalizados con el desplazamiento vertical producido en el eje sin fisura, con éste sometido a un esfuerzo de flexión.

α	β = 0	β = 0.1	β = 0.2	β = 0.3	β = 0.4
0.05	0.982	0.983	0.982	0.982	0.982
0.1	0.983	0.985	0.982	0.983	0.982
0.15	0.985	0.986	0.983	0.985	0.983
0.2	0.988	0.988	0.986	0.986	0.986
0.25	0.991	0.992	0.989	0.991	0.989
0.3	0.997	0.997	0.995	0.995	0.994
0.35	1.004	1.004	1.001	1.001	1.000
0.4	1.013	1.013	1.010	1.010	1.009
0.45	1.025	1.027	1.024	1.024	1.021
0.5	1.043	1.045	1.042	1.040	1.037

Tabla 5.3a: *valores del desplazamiento vertical máximo del eje adimensionalizados en flexión (con β desde 0 hasta 0.4)*

α	β = 0.5	β = 0.6	β = 0.7	β = 0.8	β = 0.9	β = 1
0.05	0.982	0.983	0.983	0.985	0.985	0.986
0.1	0.982	0.983	0.985	0.985	0.986	0.986
0.15	0.983	0.985	0.985	0.986	0.986	0.988
0.2	0.986	0.986	0.986	0.986	0.988	0.988
0.25	0.989	0.989	0.989	0.989	0.989	0.989
0.3	0.994	0.994	0.992	0.992	0.992	0.992
0.35	1.000	0.998	0.998	0.997	0.997	0.995
0.4	1.007	1.007	1.006	1.003	1.001	1.001
0.45	1.019	1.018	1.015	1.012	1.010	1.007
0.5	1.034	1.033	1.028	1.025	1.021	1.016

Tabla 5.3b: *valores del desplazamiento vertical máximo del eje adimensionalizados en flexión (con β desde 0.5 hasta 1)*

5.1.3 Flexo-tracción

Valores de la apertura longitudinal de la fisura adimensionalizados con el valor del desplazamiento longitudinal del eje sin fisura, cuando este está sometido tanto a flexión como a tracción.

α	β =0	β =0.1	β =0.2	β =0.3	β =0.4
0.05	0.989	0.991	0.991	0.991	0.992
0.1	0.980	0.979	0.980	0.982	0.981
0.15	0.969	0.971	0.970	0.972	0.973
0.2	0.959	0.958	0.960	0.959	0.962
0.25	0.943	0.945	0.945	0.948	0.948
0.3	0.928	0.926	0.929	0.928	0.933
0.35	0.903	0.906	0.905	0.909	0.910
0.4	0.876	0.874	0.878	0.878	0.884
0.45	0.835	0.838	0.838	0.844	0.846
0.5	0.786	0.785	0.790	0.792	0.800

Tabla 5.4a: *valores de la apertura longitudinal de la fisura adimensionalizados en flexo-tracción (con β desde 0 hasta 0.4)*

α	β =0.5	β =0.6	β =0.7	β =0.8	β =0.9	β =1
0.05	0.992	0.993	0.993	0.993	0.994	0.994
0.1	0.982	0.983	0.985	0.986	0.987	0.988
0.15	0.974	0.976	0.977	0.978	0.978	0.979
0.2	0.964	0.965	0.966	0.968	0.970	0.972
0.25	0.950	0.954	0.957	0.959	0.962	0.965
0.3	0.936	0.938	0.941	0.945	0.949	0.942
0.35	0.914	0.920	0.926	0.931	0.936	0.941
0.4	0.889	0.894	0.900	0.908	0.915	0.922
0.45	0.853	0.863	0.872	0.882	0.893	0.903
0.5	0.808	0.817	0.828	0.842	0.856	0.870

Tabla 5.4b: *valores de la apertura longitudinal de la fisura adimensionalizados (flexo-tracción)*

Valores del desplazamiento vertical máximo del eje adimensionalizados con el desplazamiento vertical del eje sin fisura, cuando este está sometido tanto a esfuerzos de flexión como de tracción.

α	β =0	β =0.1	β =0.2	β =0.3	β =0.4
0.05	0.999	0.998	1	0.999	1
0.1	0.997	0.996	0.998	0.998	0.999
0.15	0.993	0.992	0.995	0.994	0.996
0.2	0.987	0.986	0.989	0.989	0.990
0.25	0.978	0.977	0.980	0.980	0.982
0.3	0.966	0.965	0.968	0.968	0.971
0.35	0.948	0.948	0.951	0.952	0.955
0.4	0.924	0.924	0.927	0.929	0.933
0.45	0.892	0.892	0.895	0.898	0.903
0.5	0.847	0.847	0.851	0.854	0.860

Tabla 5.5a: *valores del desplazamiento vertical máximo del eje adimensionalizados en flexo-tracción (con β desde 0 hasta 0.4)*

α	β =0.5	β =0.6	β =0.7	β =0.8	β =0.9	β =1
0.05	1	0.998	0.999	0.998	0.998	0.996
0.1	0.998	0.999	0.998	0.998	0.997	0.996
0.15	0.996	0.997	0.996	0.996	0.996	0.994
0.2	0.991	0.992	0.992	0.993	0.993	0.992
0.25	0.984	0.986	0.986	0.988	0.988	0.988
0.3	0.973	0.976	0.977	0.980	0.982	0.982
0.35	0.958	0.962	0.965	0.968	0.972	0.974
0.4	0.937	0.942	0.947	0.953	0.958	0.962
0.45	0.908	0.915	0.922	0.930	0.939	0.945
0.5	0.867	0.877	0.887	0.898	0.911	0.921

Tabla 5.5b: *valores del desplazamiento vertical máximo del eje adimensionalizados en flexo-tracción (con β desde 0.5 hasta 1)*

5.1.4 Cálculo del área de la fisura

Como ya se ha explicado, resolviendo el problema inverso se pretenden obtener tres parámetros característicos del frente de la fisura a partir del valor de la apertura longitudinal de la misma o del desplazamiento vertical del eje, según corresponda.

Tanto la longitud característica de la fisura (α), como el factor de forma (β) son valores que se han asignado previamente (apartado 5.1, págs. 26 y 27 de este documento), por lo que se debe calcular el valor del área de la fisura en tanto por uno del área total del eje para cada pareja de valores de α y β. Los valores de este parámetro no dependerán de las cargas que estén aplicadas en el eje, por lo que los resultados serán iguales para los tres casos.

α	**β =0**	**β =0.1**	**β =0.2**	**β =0.3**	**β =0.4**
0.05	1.87E-02	1.78E-02	1.56E-02	1.31E-02	1.09E-02
0.1	0.052	5.09E-02	4.78E-02	4.33E-02	3.82E-02
0.15	9.41E-02	9.28E-02	8.91E-02	8.34E-02	7.65E-02
0.2	0.1423	1.41E-01	1.37E-01	1.31E-01	1.23E-01
0.25	0.1954	1.94E-01	1.90E-01	1.84E-01	1.75E-01
0.3	0.2523	2.51E-01	2.47E-01	2.40E-01	2.31E-01
0.35	0.3119	3.11E-01	3.07E-01	3.00E-01	2.91E-01
0.4	0.3736	3.72E-01	3.68E-01	3.62E-01	3.54E-01
0.45	0.4364	4.35E-01	4.32E-01	4.26E-01	4.18E-01
0.5	0.5	4.99E-01	4.96E-01	4.90E-01	4.83E-01

Tabla 5.6a: *valores del área de la fisura en tanto por uno del área total de la sección del eje (con β desde 0 hasta 0.4)*

α	β =0.5	β =0.6	β =0.7	β =0.8	β =0.9	β =1
0.05	9.17E-03	7.89E-03	6.88E-03	5.89E-03	5.47E-03	4.84E-03
0.1	3.34E-02	2.95E-02	2.61E-02	2.33E-02	2.10E-02	1.91E-02
0.15	6.93E-02	6.23E-02	5.60E-02	5.07E-02	4.61E-02	4.21E-02
0.2	1.14E-01	1.04E-01	9.51E-02	8.68E-02	7.96E-02	7.32E-02
0.25	1.64E-01	1.53E-01	1.42E-01	1.31E-01	1.21E-01	1.12E-01
0.3	2.20E-01	2.08E-01	1.94E-01	1.81E-01	1.69E-01	1.57E-01
0.35	2.80E-01	2.67E-01	2.52E-01	2.37E-01	2.22E-01	2.08E-01
0.4	3.42E-01	3.29E-01	3.14E-01	2.98E-01	2.81E-01	2.65E-01
0.45	4.07E-01	3.94E-01	3.79E-01	3.62E-01	3.44E-01	3.26E-01
0.5	4.73E-01	4.60E-01	4.46E-01	4.29E-01	4.10E-01	3.96E-01

Tabla 5.6b: *valores del área de la fisura en tanto por uno del área total de la sección del eje (con β desde 0.5 hasta 1)*

Debe resaltarse que en el Proyecto Fin de Carrera de María Ruiz Ayuso [3], se estudió mediante análisis de elementos finitos el comportamiento de una fisura en un eje sometido a esfuerzos de flexión de manera similar a como se realizaron los análisis en este proyecto.

Una vez calculados todos los datos necesarios para los tres casos, se puede proceder a realizar las redes neuronales correspondientes.

5.2 Programación de las redes neuronales

Como se ha expuesto en el aparatado 1.2 de este documento, el objetivo de este proyecto es resolver el problema inverso de detección e identificación de fisuras de frente elíptico en un eje de sección circular, mediante la aplicación de Redes Neuronales Artificiales (RNA).

De acuerdo a lo descrito en el capítulo 4, hay diferentes tipos de RNA según sea su arquitectura y el aprendizaje empleado. En este caso las RNA que mejor se adaptan al problema son las redes neuronales perceptron multicapa con algoritmo de retropropagación, sometidas a un aprendizaje supervisado, debido a que se hará coincidir cada entrada de la red con una salida específica. Se tomarán como entradas los valores del desplazamiento longitudinal o del desplazamiento vertical del eje, según corresponda y como salidas los diferentes parámetros característicos de la fisura.

Además se ha decidido desarrollar una red neuronal específica para estimar cada uno de los parámetros característicos de la fisura, cuyos códigos están recopilados en el anexo que puede encontrarse al final de este documento. Esto es debido a que se realizó una única red para estimar todos los parámetros a la vez y los resultados obtenidos fueron peores que los obtenidos de la estimación individual.

Para el desarrollo de cada una de las diferentes redes se procederá a seguir los siguientes pasos:

1. Recopilación de los datos de salida y entrada para la red.
2. Adecuación de la estructura de la red a los diferentes tipos de datos.
3. Obtención de la red entrenada.

A continuación se describen con mayor detalle cada uno de los pasos.

5.2.1 Recopilación de datos

Lo primero que se necesita es un archivo de texto con los diferentes datos que se quieren introducir en la red para que ésta realice una estimación. En este caso se realizará un aprendizaje supervisado de las redes, por lo que en el archivo se deben incluir, tanto los datos de salida como los de entrada.

En todos los casos se tomarán como entradas los valores de la apertura longitudinal de la fisura o el desplazamiento vertical del eje, según corresponda, los cuales se encuentran recopilados en las tablas del apartado anterior (tablas 5.2, 5.3, 5.4 y 5.5) y como salidas los valores de los parámetros característicos de la fisura: la longitud característica (α), el factor de forma (β) y el área de la fisura (tabla 5.6), todos ellos adimensionalizados.

Todas las redes que utilizan un algoritmo de retropropagación realizan un test de validación durante el proceso de aprendizaje mediante el cual se calcula el error cometido en la señal de salida para su posterior propagación hacia atrás por toda la red. Después de tener la red ya entrenada se debe realizar la generalización con un grupo de datos que no han sido usados durante el aprendizaje. Por ello se dividirá el total de los datos en dos grupos. Usualmente el 90% de los datos se utiliza para el aprendizaje de la red y el 10% restante se destina al test de validación.

En este proyecto se han recopilado en total 110 valores de cada uno de los parámetros sometidos a estudio para cada caso comprendido en este análisis: tracción, flexión y la combinación de ambas. De estos 110 valores se destinarán aleatoriamente 90 para el aprendizaje de la red y los 20 restantes para la generalización posterior.

Una vez que ya se tienen los archivos de datos, tanto para el entrenamiento de la red como para la generalización, se deberá indicar en el código de la red qué datos corresponden a las entradas y cuáles las salidas.

5.2.2 Adecuación de la estructura de la red

Partiendo del código básico de una red neuronal perceptron multicapa, en concreto dos capas, con algoritmo de retropropagación y utilizando como función de activación la función sigmoidea o logística citada en el apartado 4.1.3 de este documento, se procedió a

ajustar el código a los datos obtenidos. Para ello se puede modificar el número de datos que tomará la red de forma aleatoria para realizar la validación durante el entrenamiento y el número de neuronas en las capas ocultas. Estos parámetros se deben variar hasta que obtengamos unos valores óptimos de salida de la red (códigos disponibles en el anexo).

Matlab nos proporciona diferentes gráficos con los que poder observar cuánto se ajustan los datos estimados reales (figura 5.11).

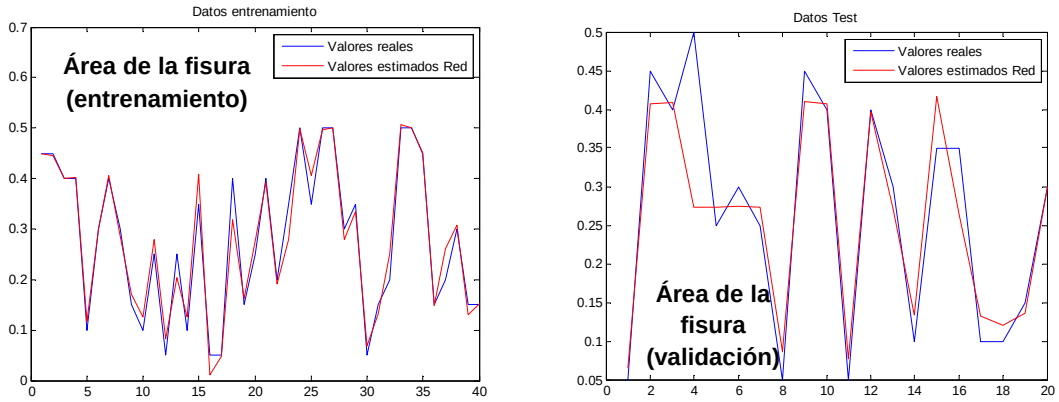


Figura 5.11: *ejemplo de representación de datos reales y estimados por la red*

Además de proporcionarnos gráficos, Matlab también ofrece la posibilidad de calcular el error cuadrático medio (ecuación 5.1) entre los valores reales que se introducen en la red y los valores que ésta estima. La fórmula utilizada para este cálculo es la siguiente:

$$E = \frac{\sum_{i=1}^{n} (est_i - real_i)^2}{n} \qquad (5.1)$$

donde (*est)* es el valor estimado por la red y (*real)* el valor introducido en la misma.

5.2.3 Obtención de la red entrenada

Una vez que se ha verificado que los valores estimados por la red son óptimos y que se ajustan a los valores reales que se han introducido, se puede decir que la red está entrenada de forma correcta.

La red ya entrenada puede utilizarse para estimar los datos de salida. Introduciendo las entradas, es decir, introduciendo los valores de la apertura longitudinal de la fisura o del desplazamiento vertical del eje, se pueden conocer los parámetros característicos: factor de forma, longitud característica y área de la fisura.

Por lo justificado en el aptdo. 5.2 (pág. 40) del presente documento, se han realizado redes neuronales para la estimación de cada uno de los parámetros característicos de la fisura en cada uno de los casos sometidos a estudio. A continuación se mostrarán los gráficos obtenidos durante el aprendizaje de las redes, los cuales muestran los datos reales frente a los estimados durante el entrenamiento. En todos los casos en el eje de ordenadas se representan los valores que toma el parámetro y en el eje de abcisas el número de datos utilizados para realizar el test de validación durante el entrenamiento.

5.2.3.1 *Tracción*

- Representación de los datos obtenidos mediante una red neuronal perceptron multicapa con algoritmo de retropropagación (figura 5.12), para la estimación de la longitud característica de la fisura (α): en este caso se han establecido como entradas de la red los valores de la apertura longitudinal de la fisura adimensionalizados y como salidas los valores del parámetro α (ver tablas 5.2).

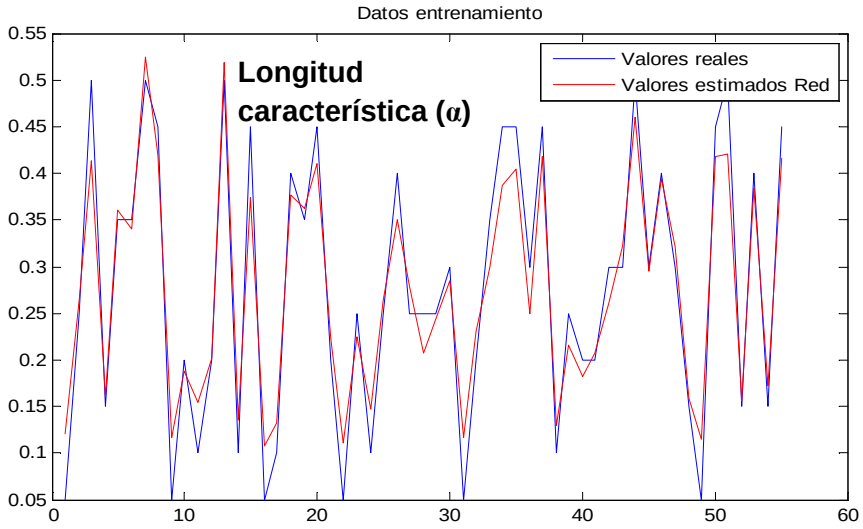


Figura 5.12: *valores de α reales y estimados por la red (entrenamiento tracción)*

- Representación de los datos obtenidos mediante una red neuronal perceptron multicapa con algoritmo de retropropagación (figura 5.13), para la estimación del factor de forma de la fisura (β): se han establecido como entradas de la red los valores de la apertura longitudinal de la fisura adimensionalizados y como salidas los valores del parámetro β (ver tablas 5.2).

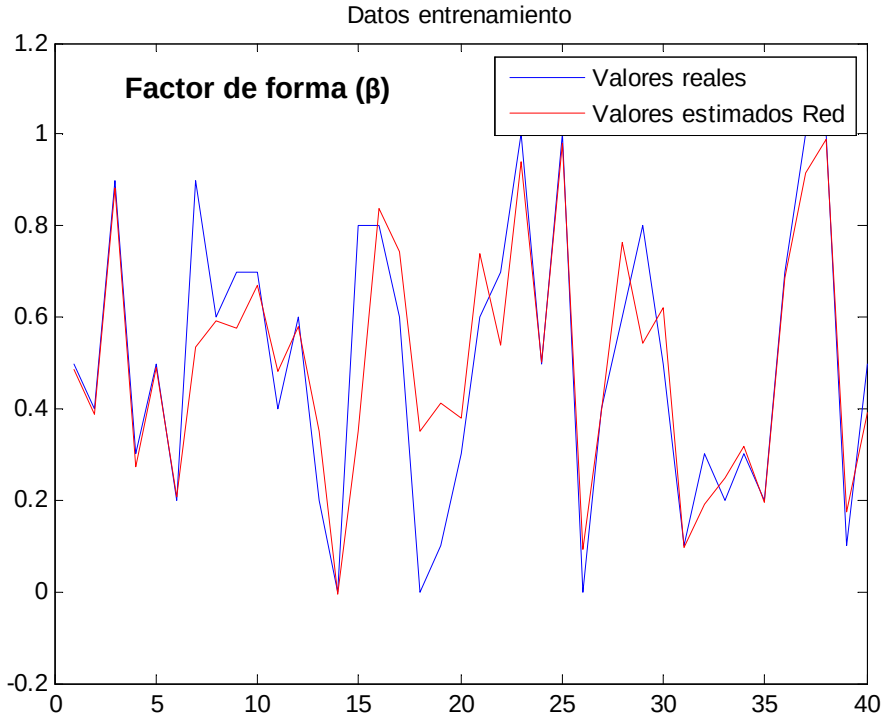


Figura 5.13: *valores de β reales y estimados por la red (entrenamiento tracción)*

- Representación de los datos obtenidos mediante una red neuronal perceptron multicapa con algoritmo de retropropagación (figura 5.14), para la estimación del área de la fisura en tanto por uno del área total de la sección del eje: en este caso se han establecido como entradas de la red los valores de la apertura longitudinal de la fisura adimensionalizados (ver tablas 5.2) y como salidas los valores calculados del área de la fisura en tanto por uno del área total del eje (ver tablas 5.6).

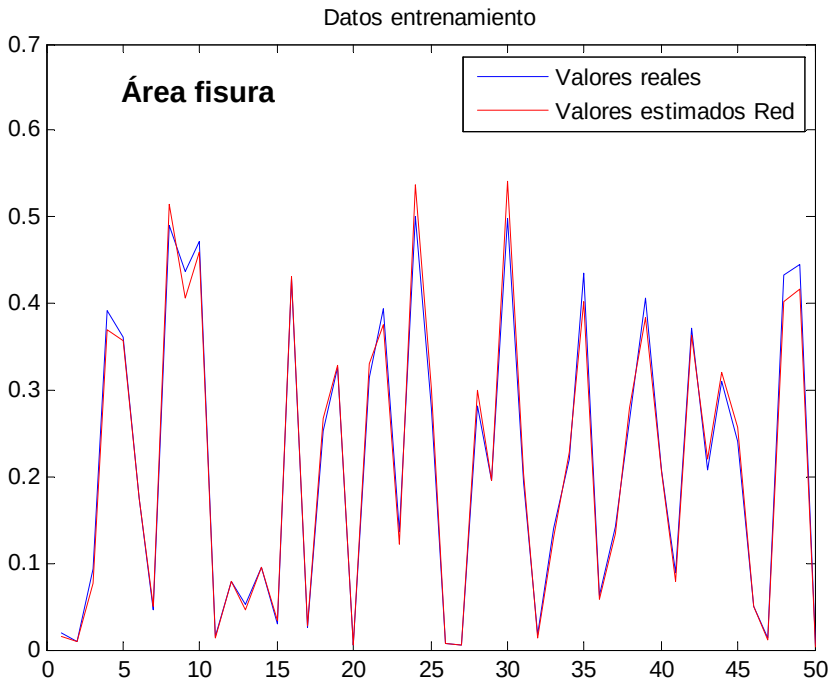


Figura 5.14: *valores del área de la fisura reales y estimados por la red (entrenamiento tracción)*

5.2.3.2 *Flexión*

- Representación de los datos obtenidos mediante una red neuronal perceptron multicapa con algoritmo de retropropagación (figura 5.15), para la estimación de la longitud característica de la fisura (α): en este caso se han establecido como entradas de la red los valores del desplazamiento vertical del eje adimensionalizados y como salidas los valores del parámetro α (ver tablas 5.3).

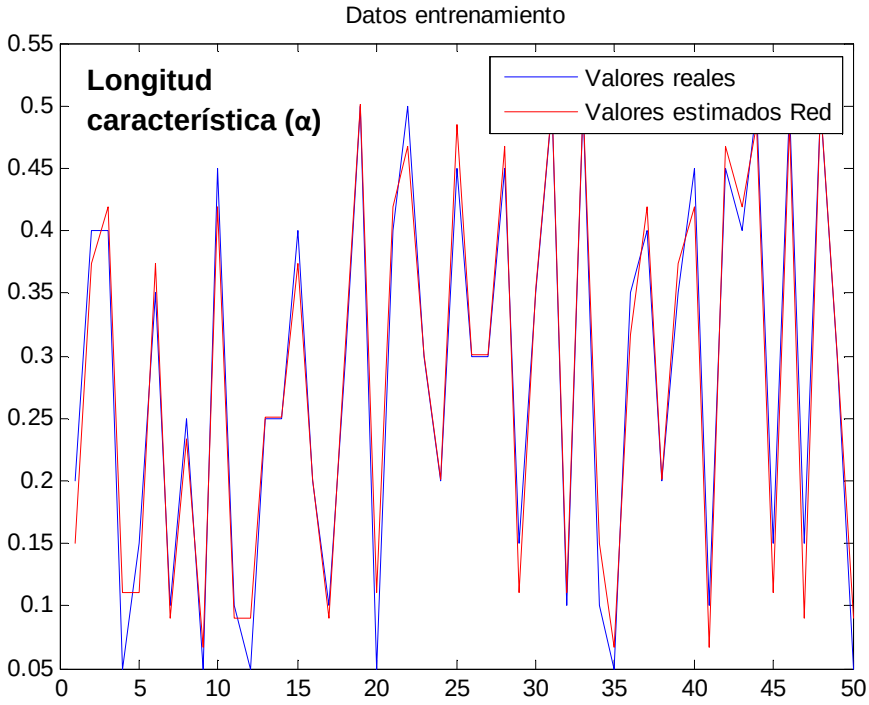


Figura 5.15: *valores de α reales y estimados por la red (entrenamiento flexión)*

- Representación de los datos obtenidos mediante una red neuronal perceptron multicapa con algoritmo de retropropagación (figura 5.16), para la estimación del factor de forma de la fisura (β): en este caso se han establecido como entradas de la red los valores del desplazamiento vertical del eje adimensionalizados y como salidas los valores del parámetro β (ver tablas 5.3).

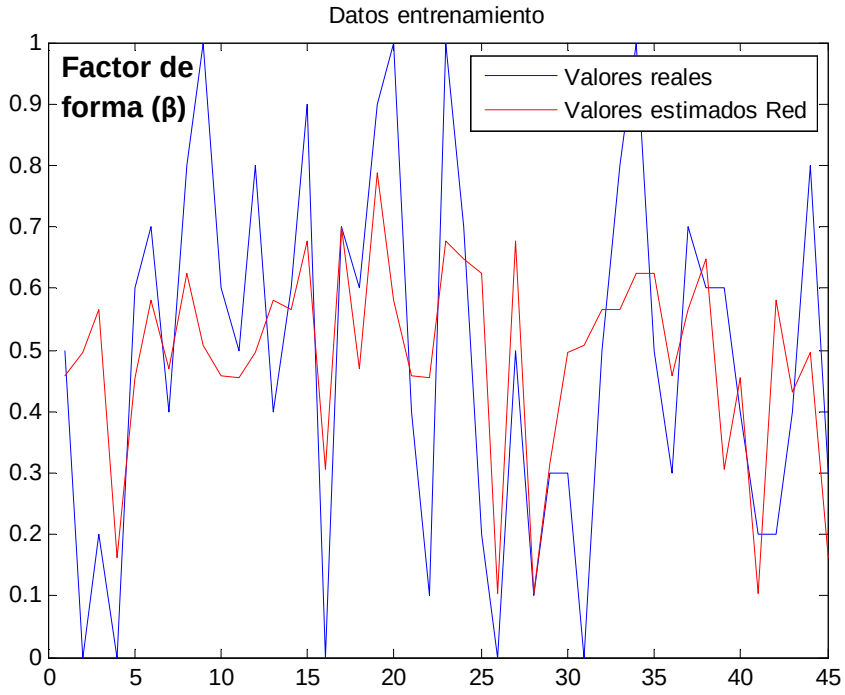


Figura 5.16: *valores de β reales y estimados por la red (entrenamiento flexión)*

- Representación de los datos obtenidos mediante una red neuronal perceptron multicapa con algoritmo de retropropagación (figura 5.17), para la estimación del área de la fisura en tanto por uno del área total del eje: en este caso se han establecido como entradas de la red los valores del desplazamiento vertical del eje adimensionalizados (ver tablas 5.3) y como salidas los valores calculados del área de la fisura en tanto por uno del área total del eje (ver tablas 5.6).

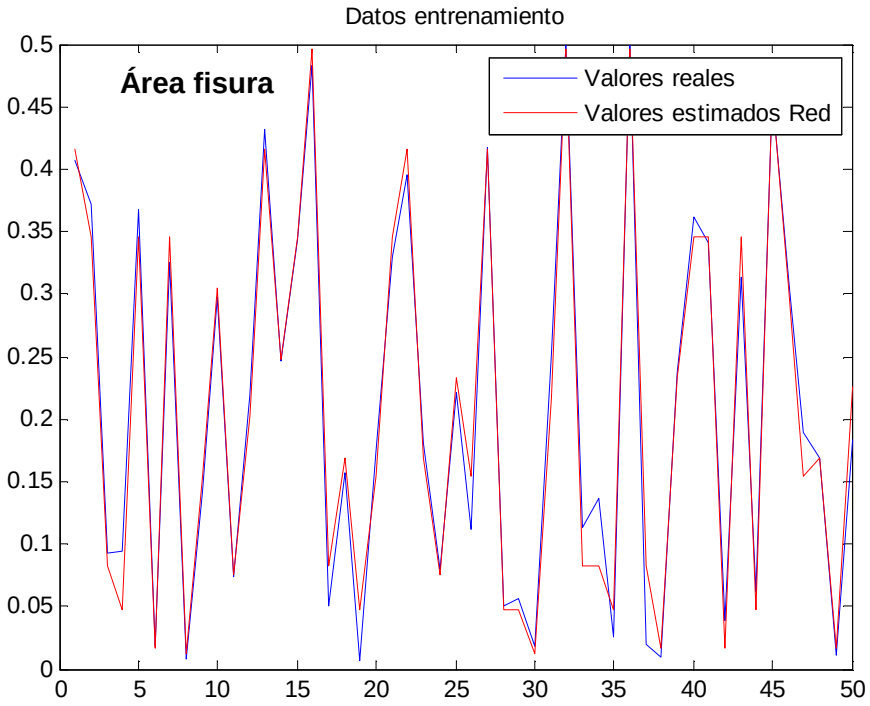


Figura 5.17: *valores del área de la fisura reales y estimados por la red (entrenamiento flexión)*

5.2.3.3 *Flexo-tracción*

En el caso de tracción combinada con flexión aparecen tanto incrementos de longitud horizontales como verticales y por lo tanto, se deben analizar los parámetros característicos para ambos.

- Representación de los datos obtenidos mediante una red neuronal perceptron multicapa con algoritmo de retropropagación (figura 5.18), para la estimación de la longitud característica de la fisura (α): en este caso se han establecido como entradas de la red los valores del desplazamiento vertical del eje y los valores de la apertura longitudinal de la fisura, todos ellos adimensionalizados y como salidas los valores del parámetro α (ver tablas 5.4 y 5.5).

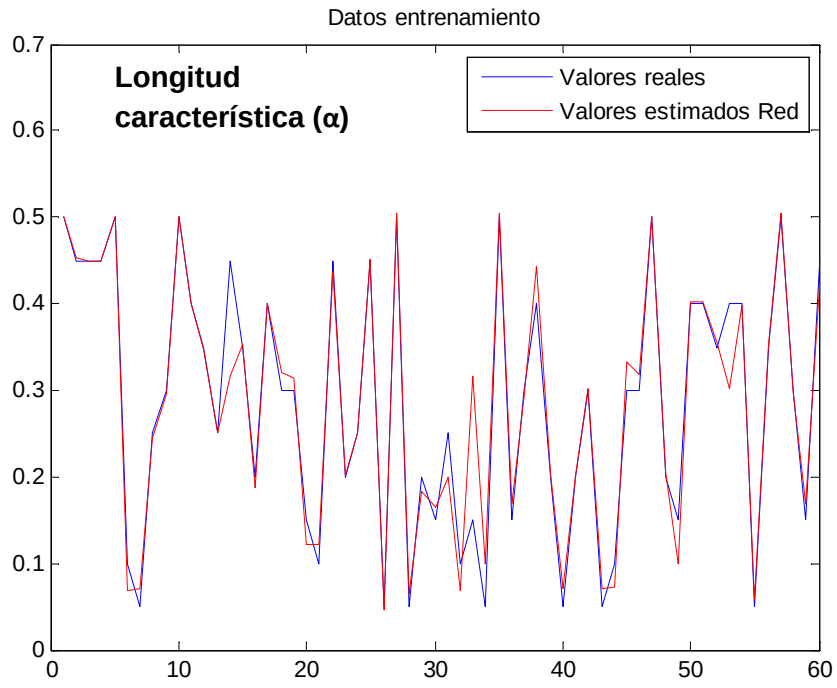


Figura 5.18: *valores de α reales y estimados por la red (entrenamiento flexo-tracción)*

- Representación de los datos obtenidos mediante una red neuronal perceptron multicapa con algoritmo de retropropagación (figura 5.19), para la estimación del factor de forma de la fisura (β): en este caso se han establecido como entradas de la red los valores del desplazamiento vertical del eje y los valores de la apertura longitudinal de la fisura, todos ellos adimensionalizados y como salidas los valores del parámetro β (ver tablas 5.4 y 5.5).

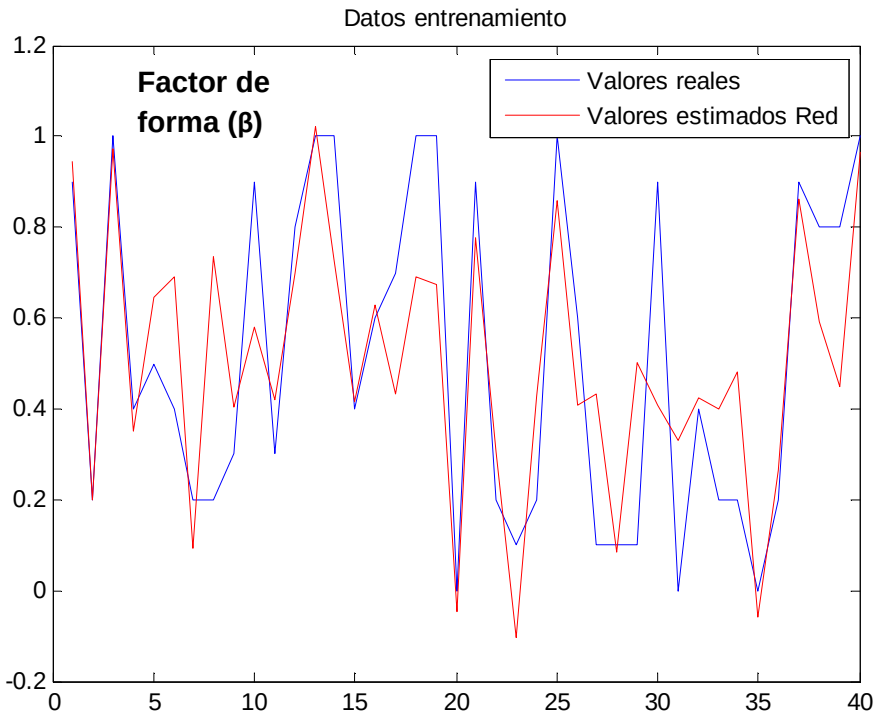


Figura 5.19: *valores de β reales y estimados por la red (entrenamiento flexo-tracción)*

- Representación de los datos obtenidos mediante una red neuronal perceptron multicapa con algoritmo de retropropagación (figura 5.20), para la estimación del área de la fisura en tanto por uno del área total del eje: en este caso se han establecido como entradas de la red los valores del desplazamiento vertical del eje y los valores de la apertura longitudinal de la fisura, todos ellos adimensionalizados (ver tablas 5.4 y 5.5) y como salidas los valores calculados del área de la fisura en tanto por uno del área total de la sección del eje (ver tablas 5.6).

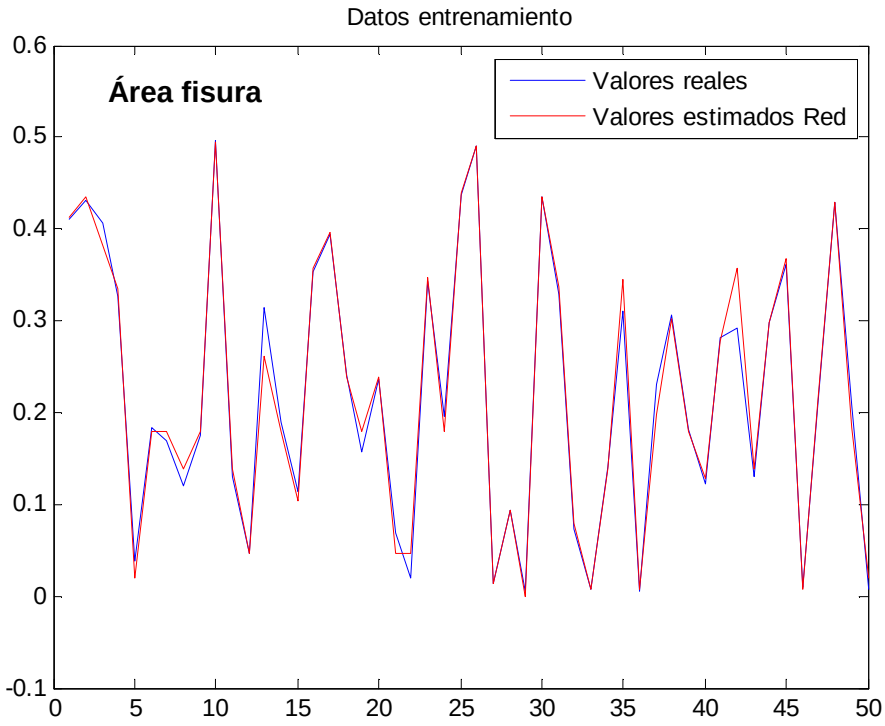


Figura 5.20: *valores del área de la fisura reales y estimados por la red (entrenamiento flexo-tracción)*

En la Tabla 5.7 se muestran los errores cuadráticos medios cometidos por la red en el cálculo de la estimación durante la etapa de entrenamiento.

Error cuadrático medio	Longitud característica (α)	Factor de forma (β)	Área de la fisura
Tracción	0.0018	0.0485	0.0018
Flexión	0.0009	0.0666	$0.5627 \cdot 10^{-3}$
Flexo-tracción	0.0023	0.0518	0.0005

Tabla 5.7: *valor del error cuadrático medio de la estimación cometido por la red en el entrenamiento*

Capítulo 6

Resultados y discusión

6.1 Resultados

Se han obtenido mediante el método de elementos finitos, los valores de los desplazamientos verticales y horizontales, en un eje de sección circular de acero que presenta una fisura de frente semielíptico, para diferentes valores de varios parámetros característicos de la fisura.

Todos estos valores se han utilizado para realizar diversas redes neuronales, con el objetivo de poder resolver con ellas el problema inverso, es decir, a partir de los valores del desplazamiento, poder estimar los valores de los parámetros característicos de la fisura. Las representaciones gráficas del aprendizaje o entrenamiento de estas redes están recogidas en el apartado 5.3 de este documento.

Estas gráficas muestran los datos reales introducidos y los datos estimados por la red. Para ver realmente si las redes están correctamente entrenadas, es decir, podrían estimar con un mínimo error los parámetros de la fisura a partir de los desplazamientos en el eje, éstas realizan un test de validación utilizando 20 datos escogidos al azar que no han intervenido en su etapa de aprendizaje. Observando los resultados obtenidos podemos conocer el error cometido a la hora de realizar las estimaciones.

Además de realizarse gráficas que muestran los datos reales frente a los estimados y calcularse el error cuadrático medio cometido por la red durante la etapa de validación (fig. 6.1, 6.3 y 6.5), se han realizado con los mismos datos unos gráficos de dispersión en los que se ha trazado una línea a 45º de los ejes (fig. 6.2, 6.4 y 6.6) con el objetivo de poder apreciar la magnitud del error cometido en el cálculo de la estimación de forma gráfica. En el caso ideal de que la red estimase los valores sin error, los puntos representados deberían situarse sobre la recta dibujada. También se ha obtenido el factor R^2 que representa el grado de ajuste de los puntos a una línea. Este valor varía de 0 a 1 y cuánto más cercano sea a la unidad, mejor será el grado de ajuste y por lo tanto, menor será el error cometido en la estimación.

A continuación se mostrarán las gráficas realizadas para los resultados del test de validación en cada uno de los casos junto con las gráficas de dispersión, con el fin de poder apreciar mejor el error cometido por las redes. En todas las gráficas realizadas por Matlab, en el eje de ordenadas se representan los valores que toma el parámetro y en el eje de abcisas el número de datos (20) sobre los que se aplicó la red ya entrenada para realizar la generalización.

6.1.1 Tracción

Representaciones gráficas de los datos reales y estimados durante el test de validación realizado por la red, de la longitud característica, factor de forma y área de la fisura en tanto por uno del área total del eje (figura 6.1) cuando este está sometido a tracción.

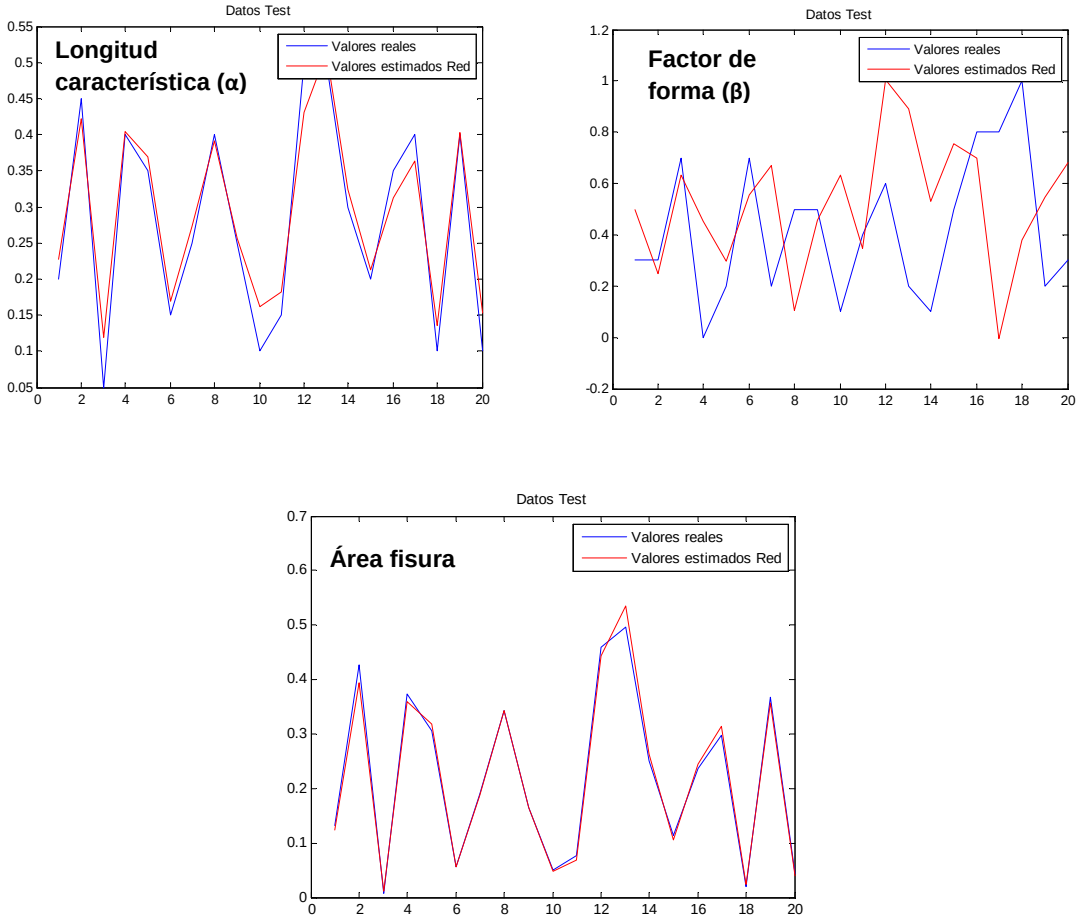


Figura 6.1: *representación datos reales y estimados en el test de validación de α, β y área de la fisura (tracción)*

Gráficos de dispersión y cálculo del factor R^2 que representa el grado de ajuste de los puntos a una línea, para los distintos parámetros de la fisura (figura 6.2), cuando el eje está sometido a un esfuerzo de tracción.

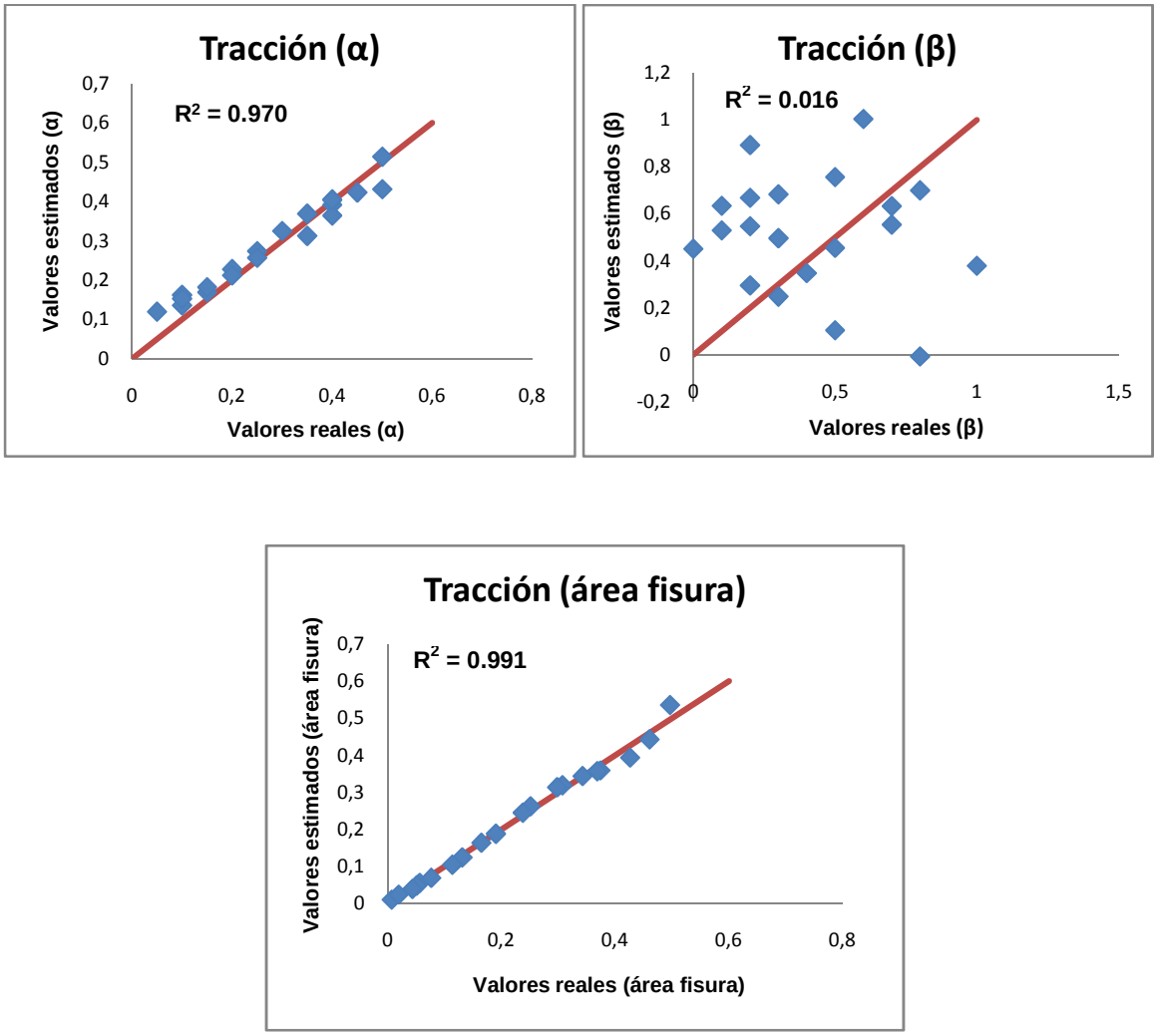


Figura 6.2: *representación del error cometido en el test de validación de α, β y el área de la fisura (tracción)*

6.1.2 Flexión

Representaciones gráficas de los datos reales y estimados durante el test de validación realizado por la red, de la longitud característica (α), factor de forma (β) y área de la fisura en tanto por uno del área total del eje (figura 6.3), cuando este está sometido a flexión.

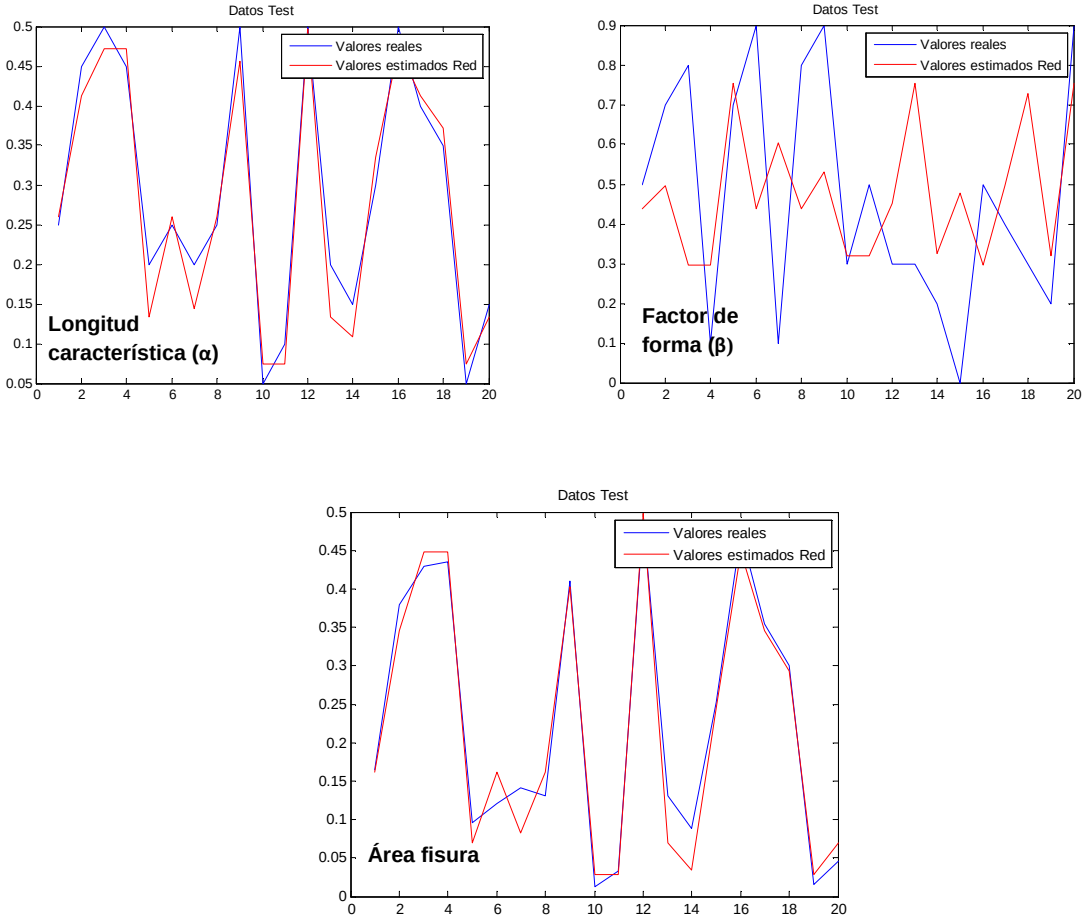


Figura 6.3: *representación datos reales y estimados en el test de validación de α, β y área de la fisura (flexión)*

Gráficos de dispersión y cálculo del factor R^2 que representa el grado de ajuste de los puntos a una línea, para los distintos parámetros de la fisura (figura 6.a), cuando el eje está sometido a un esfuerzo de flexión.

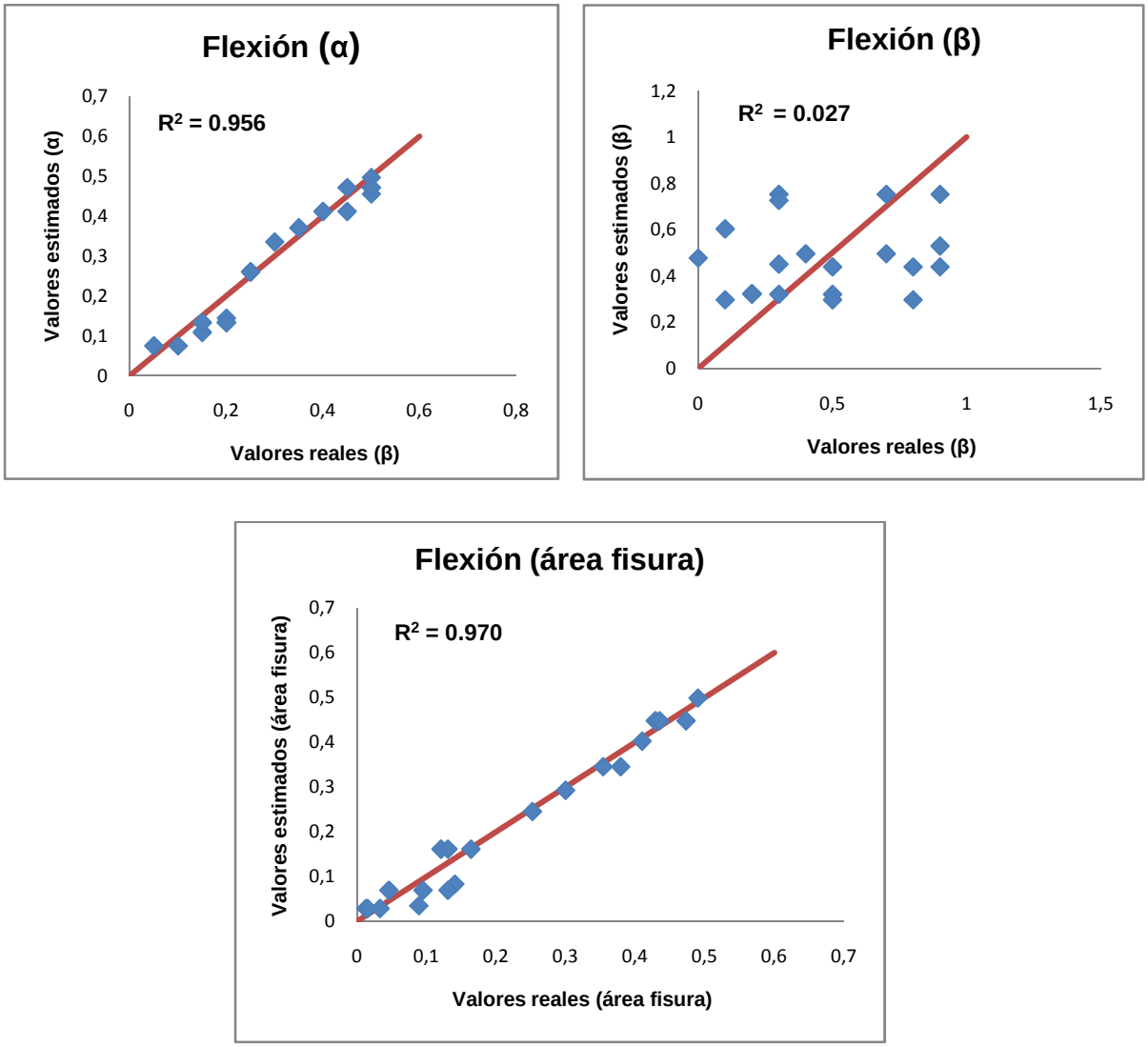


Figura 6.4: *representación de los errores cometidos en el test de validación mediante un ajuste lineal de los datos reales y estimados de α, β y el área de la fisura (flexión)*

6.1.3 Flexo-tracción

Representaciones gráficas de los datos reales y estimados durante el test de validación realizado por la red, de la longitud característica (α), factor de forma (β) y área de la fisura en tanto por uno del área total del eje (figura 6.5), cuando este está sometido a flexo-tracción.

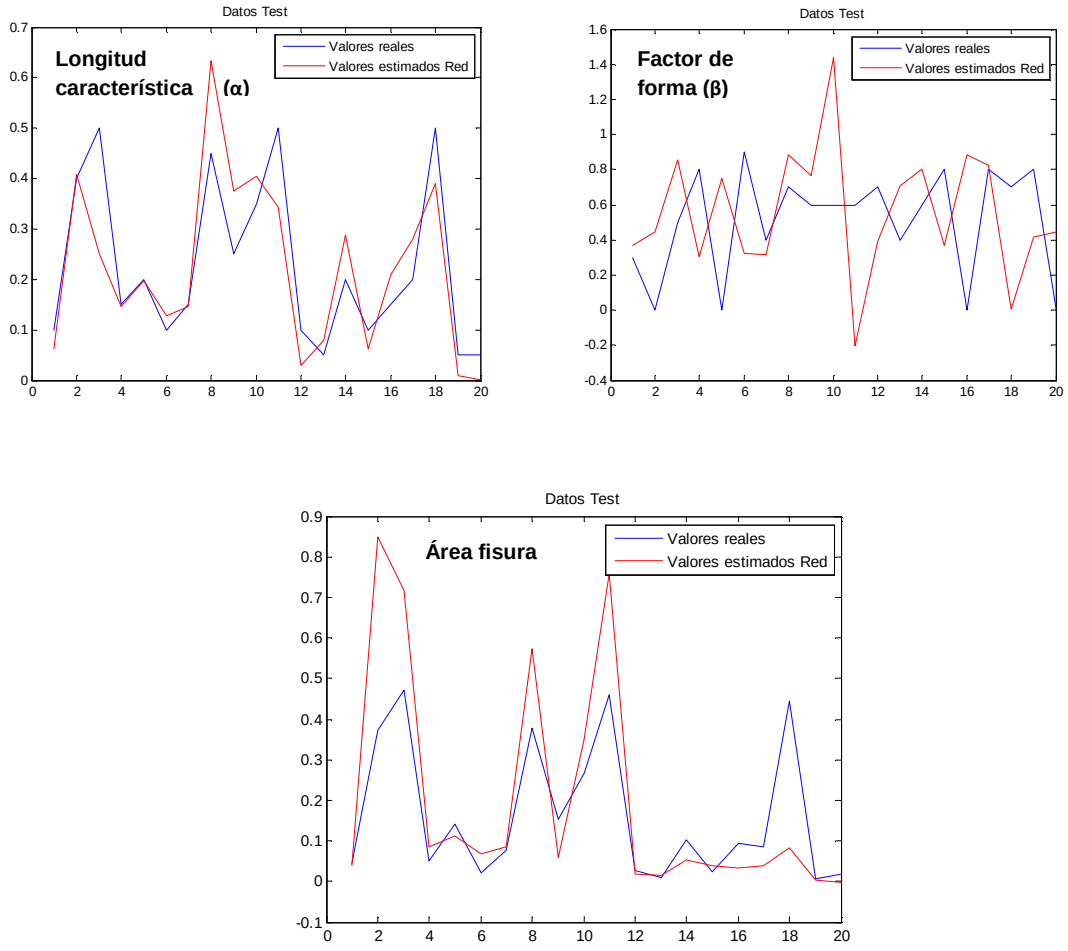


Figura 6.5: *representación datos reales y estimados en el test de validación de α, β y área de la fisura (flexo-tracción)*

Gráficos de dispersión y cálculo del factor R^2 que representa el grado de ajuste de los puntos a una línea, para los distintos parámetros de la fisura (figura 6.6), cuando el eje está sometido a esfuerzos de tracción y flexión.

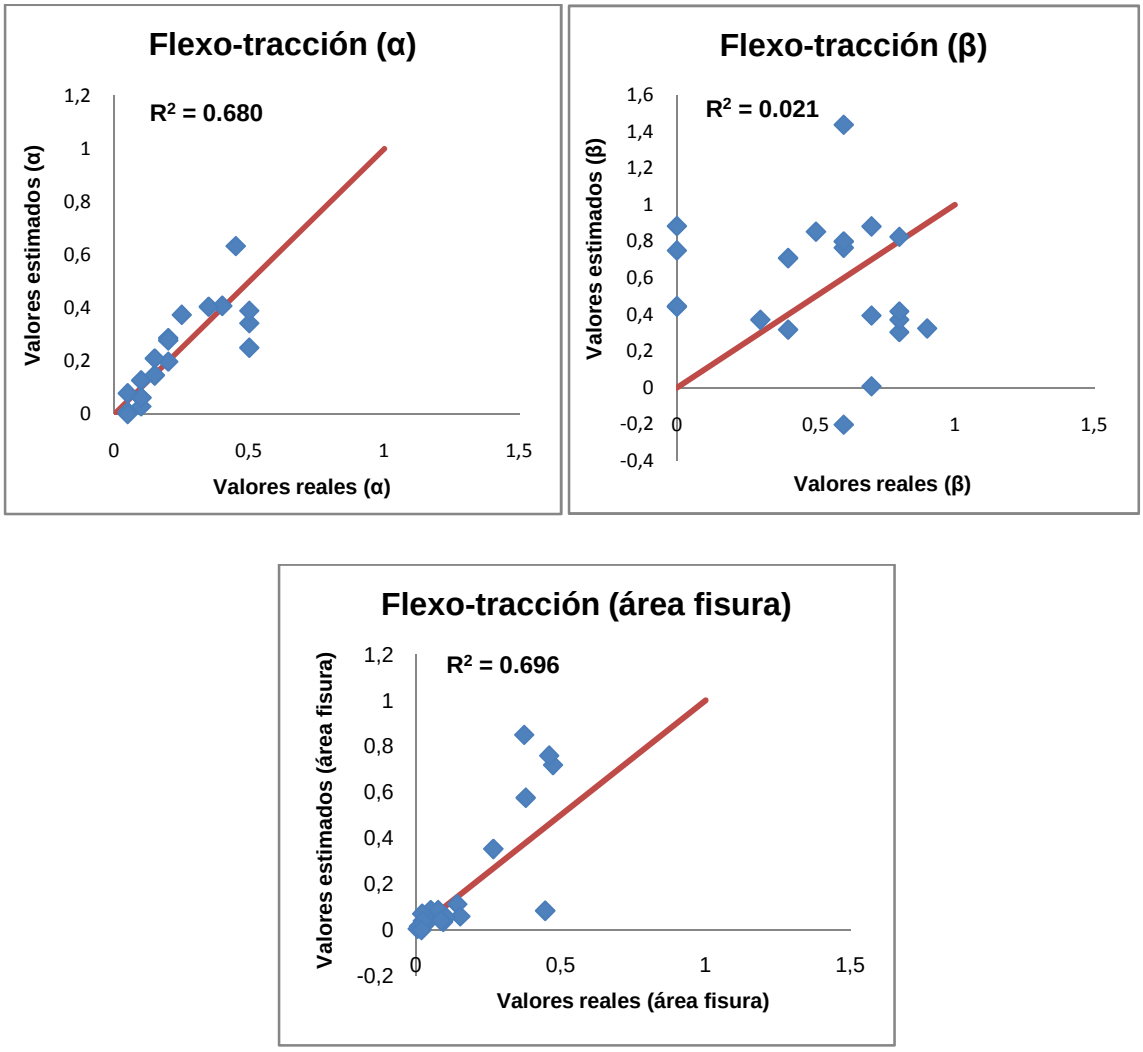


Figura 6.6: *representación de los errores cometidos en el test de validación mediante un ajuste lineal de los datos reales y estimados de α, β y el área de la fisura (flexo-tracción)*

En la tabla 6.1 se muestran los valores calculados del error cuadrático medio (ECM) para la estimación realizada por la red en el test de validación, para cada uno de los casos y parámetros de interés. En la tabla 6.2 se recogen los valores del grado de ajuste de los puntos a una recta (R^2) para los gráficos de dispersión de los datos reales y estimados del test de validación.

ECM para test validación	Longitud característica (α)	Factor de forma (β)	Área de la fisura
Tracción	0.0013	0.1576	$0.2028 \cdot 10^{-3}$
Flexión	0.0012	0.0923	$0.8460 \cdot 10^{-3}$
Flexo-tracción	0.0091	0.2448	0.0288

Tabla 6.1: *ECM de la estimación de α, β y el área de la fisura para el test de validación*

R^2 para test validación	Longitud característica (α)	Factor de forma (β)	Área de la fisura
Tracción	0.970	0.016	0.991
Flexión	0.956	0.027	0.970
Flexo-tracción	0.680	0.021	0.696

Tabla 6.2: *R^2 para los valores reales y estimados de α, β y el área de la fisura utilizados en el test de validación*

De los resultados se puede extraer que el área de la fisura es el parámetro mejor estimado por las redes, mientras que el factor de forma (β) es el parámetro cuyo error de estimación es mayor para los tres casos. Además el error cometido al estimar los parámetros en el caso de flexo-tracción es mayor que para tracción y flexión por separado. Los valores del grado de ajuste de los puntos a una recta (R^2) mostrados en las gráficas de dispersión del apartado anterior corroboran las conclusiones anteriores (figuras 6.2, 6.4 y 6.6).

6.2 Discusión de los resultados.

A la vista de los resultados mostrados en el apartado 6.1 se puede extraer que todos los parámetros no se pueden estimar con la misma precisión, ya que los errores de estimación varían. Además no solo influye el parámetro utilizado sino que también influye el esfuerzo al que está sometido el eje.

Como muestra la tabla 6.1, los errores cuadráticos medios calculados para los 20 valores sobre los que se realizó el test de validación una vez que la red ya estaba entrenada, varían para cada uno de los parámetros característicos de la fisura sometidos a estudio.

- *Longitud característica* (α). El error cuadrático medio cometido al estimar α es pequeño y del mismo orden para los tres casos (tracción, flexión y flexo-tracción), lo que demuestra que este parámetro puede ser estimado por las redes satisfactoriamente.

- *Factor de forma* (β). El error cuadrático medio de la estimación es también del mismo orden, pero en este caso es mayor por lo que se puede extraer que en el caso de β la estimación con redes neuronales no sería viable. La información obtenida del grado de ajuste lineal en las gráficas de dispersión (R^2) recogida en la tabla 6.2 ofrece la misma conclusión.

- *Área de la fisura en tanto por uno del área total del eje*. A la vista de los resultados, este parámetro es el que ofrece los resultados más satisfactorios a la hora de realizar su estimación. El error cuadrático medio calculado para el test de validación en los tres casos (tabla 6.1) es de un orden bajo, aunque para el caso de esfuerzos combinados puede observarse que la estimación resulta menos precisa.

De todo lo anterior puede resumirse que tanto la longitud característica como el área de la fisura son parámetros que pueden estimarse a través de redes neuronales partiendo de los desplazamientos sufridos en el eje. Pero hay que matizar que los resultados para los casos en los que el eje está sometido a un solo tipo de esfuerzo (tracción o flexión simple) ofrecen errores de estimación menores que en el caso en el que el eje se encuentra bajo esfuerzos combinados de flexión y tracción, sobre para fisuras grandes.

En lo referente al factor de forma, se puede concluir que su estimación mediante una red neuronal perceptron multicapa con algoritmo de retropropagación como las utilizadas en este proyecto no sería viable, debido a que el error cometido en la estimación es mayor que para los otros parámetros y sus resultados no serían fieles a la realidad.

De este hecho se deduce que los valores del desplazamiento en un eje de sección circular que presenta una fisura de frente elíptico, tienen una relación más fácil de estimar por la red con respecto a la longitud característica (α) y el área de la fisura que con el factor de forma (β) de la misma. Esto es debido a que resulta muy complejo estimar β al tratarse de una fisura que tiene el frente de forma elíptica y no de otra forma más sencilla, como se asume en las simplificaciones más comunes, en las que se supone que la fisura tiene el frente recto.

La conclusión final extraída de los resultados obtenidos es que, de acuerdo a los errores de estimación cometidos por las redes neuronales, tanto la longitud característica como el área de la fisura pueden ser calculados mediante la solución del problema inverso, en el cuál se obtienen los valores de ambos parámetros a partir de la apertura longitudinal y el desplazamiento vertical del eje, estando éste sometido a diferente esfuerzos. En el caso del factor de forma, debido a que la fisura que estamos tratando tiene el frente de forma elíptica, no sería posible su estimación ya que los resultados calculados por la red no se ajustan a los datos reales.

También debe resaltarse que los resultados obtenidos cuando el eje está sometido a esfuerzos combinados de tracción y flexión para fisuras grandes, son menos precisos que los resultados para la estimación de los mismos parámetros cuando el eje está sometido a solo uno de los esfuerzos, como demuestran los valores de los errores calculados y las gráficas de dispersión realizadas.

Capítulo 7

Conclusiones y trabajos futuros

7.1 Conclusiones

En este proyecto se han utilizado redes neuronales artificiales para poder estimar los parámetros característicos de una fisura de frente elíptico en un eje de acero de sección circular, sometido a diferentes combinaciones de esfuerzos.

En la primera parte del documento se han procedido a calcular los distintos desplazamientos producidos en un eje que presenta una fisura mediante el método de elementos finitos, utilizando el programa ABAQUS, para diferentes combinaciones de valores de la longitud característica (α) y el factor de forma (β) de la fisura (tablas 5.2, 5.3, 5.4 y 5.5). El otro parámetro calculado para su posterior estimación es el área de la fisura (tablas 5.6) Estos cálculos se han realizado para tres combinaciones diferentes de esfuerzos: tracción, flexión y la combinación de ambas.

En la segunda parte, se han diseñado diferentes redes neuronales artificiales perceptron multicapa con algoritmo de retropropagación, mediante las cuales se ha procedido a resolver el problema inverso al anterior, es decir, a partir de los valores de los desplazamientos se han estimado los parámetros característicos de la fisura.

De los resultados obtenidos podemos concluir que no todos los parámetros pueden estimarse mediante este procedimiento. El área y la longitud característica de la fisura sí pueden estimarse con un error pequeño, mientras que el factor de forma de la fisura no puede obtenerse de este modo.

El motivo por el que no se puede estimar el valor del factor de forma a partir del desplazamiento en la fisura es, que al tratarse ésta de una fisura de frente elíptico, es difícil para la red estimar la forma de la fisura. Sería más sencillo de calcular si la fisura tuviera frente recto.

Debe resaltarse que los valores obtenidos para el caso del eje sometido tanto a esfuerzos de tracción como de flexión para fisuras grandes, presentan un error mayor que en los casos en los que está sometido a un solo tipo de esfuerzo.

7.2 Trabajos futuros

Tras los resultados obtenidos, los trabajos futuros que pueden proponerse como continuación al presente proyecto son:

- La resolución del mismo problema pero esta vez haciendo que el eje gire sobre sí mismo, añadiendo con ello la apertura y cierre de la fisura a las diferentes solicitaciones combinadas de flexión y tracción.

- Realizar medidas experimentales sobre un eje sometido a diferentes solicitaciones, ya que los datos utilizados para este estudio fueron obtenidos mediante el método de elementos finitos, es decir, de manera teórica y así poder observar en qué medida los resultados teóricos se asemejan a la realidad.

Bibliografía

[1] L. Rubio, B. Muñoz "Determinación de la flexibilidad de ejes con fisuras de frente elíptico". *Anales de mecánica de la fractura (24)*, pp. 587-592, 2007.

[2] Isabel González Farias "Redes Neuronales Artificiales". Transparencias de la asignatura de Técnicas Avanzadas de Diseño en Ingeniería Mecánica, impartida en el Máster en Ingeniería de Máquinas y Transporte de la Universidad Carlos III. 2008.

[3] María Ruiz Ayuso, "Estudio del comportamiento a flexión de ejes con fisuras de frente semielíptico". Proyecto de fin de carrera, Ingeniería Industrial, Universidad Carlos III de Madrid, octubre de 2010.

[4] Alfredo Catalina Gallego, "Introducción a las redes neuronales artificiales". Revista electrónica del Grupo Universitario de Informática (GUI). www.gui.uva.es/revista/login/13/redesn.html (Accedido el 21/9/11).

[5] Howard Demuth, Mark Beal "Neural Network Toolbox". User's Guide version 3.0, capítulos 1, 2 y 3. (1998) The MathWorks Inc.

[6] ABAQUS User´s manual, Version 6.7. Hibbit Karlsson & Sorenssen. Inc. 1997.

[7] www.cecalc.ula.ve/documentacion/tutoriales/abaqus/ejemplos_ABAQUS.pdf (21/9/11). Ejemplos utilizando el programa Abaqus, Universidad de los Andes.

[8] B. Muñoz-Abella, L. Rubio “Detección e identificación de fisuras de frente semielíptico en ejes mediante la aplicación de algoritmos genéticos”. XVIII Congreso Nacional de Ingeniería Mecánica. España. 2010

ANEXO

Código de programación de las redes neuronales perceptron multicapa con algoritmo de retropropagación, realizadas en el toolbox de redes neuronales en Matlab.

i. Tracción

- **Red empleada para la estimación de la longitud característica de la fisura (α).**

```
clearall

datostrain=load('DatosTraccionTrain.txt'); %lee el fichero con los datos
datostest=load('DatosTraccionTest.txt');
% el fichero DatosTraccionTrain será usado para la fase de entrenamiento
% el fichero DatosTraccionTest tiene datos no usados en el entrenamiento
y
% será usado para calcular el error

[m,K]=size(datostrain); %m será número de filas y K número de columnas
[f,K]=size(datostest);

vinput= 2; %defino qué columnas contienen las variables de entrada
voutput= 3;  % columna que contiene variable salida

InputTrain=datostrain(:,vinput);
OutputTrain=datostrain(:,voutput);
[m,p]=size(InputTrain); %m número de datos y p número de variables
entrada
[m,ov]=size(OutputTrain); % m número de datos y ov número de variables
salida

InputTest=datostest(:,vinput);
OutputTest=datostest(:,voutput);

%-------------------------------------------------------------------------
-
%Para entrenar en las redes BackProp se usa un test de validación durante
```

```
%el entrenamiento. Vamos a seleccionar aleatoriamente este conjunto

nval=35; %tamaño del set de validación
ltrain=m-nval;

posicionesval(1,1)=ceil(rand(1,1)*m); %selecciona aleatoriamente la fila
que será el primer dato para set de validación
for i=2:nval
sumiguales=1;
while sumiguales>=1
posicionesval(i,1)=ceil(rand(1,1)*m);
iguales=zeros(i-1,1);
for j=i-1:-1:1
if posicionesval(i,1)==posicionesval(j,1)
iguales(j,1)=1;
else
iguales(j,1)=0;
end
end
sumiguales=sum(iguales);
end
end
posicionesval=posicionesval; %filas elegidas como datos validación

conta=0;
for i=1:m
    u=(posicionesval==i);
    u1=sum(u);
if u1>=1;
else
conta=conta+1;
posicionestrain(conta,1)=i;
end
end
posicionestrain=sort(posicionestrain); % filas para entrenamiento

% ------------------------------------------------------------------------
--
Traininput=InputTrain(posicionestrain,:)';  % estos son los datos para
entrenar, matriz dimensión p x ltrain
Trainoutput=OutputTrain(posicionestrain,:)'; % estos son los datos para
entrenar, matriz dimensión ov x ltrain

Valinput=InputTrain(posicionesval,:)'; %estos son los datos para validar
en la fase entrenamiento
Valoutput=OutputTrain(posicionesval,:)';
%------------------------------------------------------------------------
--

net= newff(minmax(Traininput),[50 1],{'logsig''purelin'},'trainrp');
%creamos la arquitectura red
VV.P=Valinput;  %asigna a la variable P de VV los datos de entrada
VV.T=Valoutput; %asigna a la variable T de VV los datos de salida

[net,tr,Y]=train(net,Traininput,Trainoutput,[],[],VV,[]); %entrenamos
usando datos entrenamiento y validación
net.trainParam  % nos da parámetros de la red entrenada
```

```
% y contiene los valores estimados de la red para los datos de
entrenamiento

% Gráfico de datos reales y estimaciones de la red neuronal
% usamos datos de entrenamiento
figure(1)
plot(Trainoutput,'b')
holdon
plot(Y,'r')
legend('Valores reales','Valores estimados Red')
title('Datos entrenamiento')
holdoff

MSEtrain=mean((Y-Trainoutput).^2); %mean square error entrenamiento

% ------------------------------------------------------------------
%Se usa la red entrenada para el conjunto de datos no usados "Test"
TestRed=sim(net,InputTest')'; %estimación de la red para datos test

MSEtest=mean((TestRed-OutputTest).^2); %mean square error en datos test

figure(2)
plot(OutputTest(:,1),'b')
holdon
plot(TestRed(:,1),'r')
legend('Valores reales','Valores estimados Red')
title('Datos Test')
holdoff

[MSEtrain MSEtest]
[TestRed OutputTest]
```

- **Red empleada para la estimación del factor de forma de la fisura(β).**

```
clearall

datostrain=load('DatosTraccionTrain.txt'); %lee el fichero con los datos
datostest=load('DatosTraccionTest.txt');
% el fichero DatosTraccionTrain será usado para la fase de entrenamiento
% el fichero DatosTraccionTest tiene datos no usados en el entrenamiento
y
% será usado para calcular el error

[m,K]=size(datostrain); %m será número de filas y K número de columnas
[f,K]=size(datostest);

vinput= 2; %defino qué columnas contienen las variables de entrada
voutput= 4;  % columna que contiene variable salida

InputTrain=datostrain(:,vinput);
OutputTrain=datostrain(:,voutput);
[m,p]=size(InputTrain); %m número de datos y p número de variables
entrada
```

```
[m,ov]=size(OutputTrain); % m número de datos y ov número de variables
salida

InputTest=datostest(:,vinput);
OutputTest=datostest(:,voutput);

%-------------------------------------------------------------------------
-
%Para entrenar en las redes BackProp se usa un test de validación durante
%el entrenamiento. Vamos a seleccionar aleatoriamente este conjunto

nval=50; %tamaño del set de validación
ltrain=m-nval;

posicionesval(1,1)=ceil(rand(1,1)*m); %selecciona aleatoriamente la fila
que será el primer dato para set de validación
for i=2:nval
sumiguales=1;
while sumiguales>=1
posicionesval(i,1)=ceil(rand(1,1)*m);
iguales=zeros(i-1,1);
for j=i-1:-1:1
if posicionesval(i,1)==posicionesval(j,1)
iguales(j,1)=1;
else
iguales(j,1)=0;
end
end
sumiguales=sum(iguales);
end
end
posicionesval=posicionesval; %filas elegidas como datos validación

conta=0;
for i=1:m
    u=(posicionesval==i);
    u1=sum(u);
if u1>=1;
else
conta=conta+1;
posicionestrain(conta,1)=i;
end
end
posicionestrain=sort(posicionestrain); % filas para entrenamiento

% -----------------------------------------------------------------------
--
Traininput=InputTrain(posicionestrain,:)';  % estos son los datos para
entrenar, matriz dimensión p x ltrain
Trainoutput=OutputTrain(posicionestrain,:)'; % estos son los datos para
entrenar, matriz dimensión ov x ltrain

Valinput=InputTrain(posicionesval,:)'; %estos son los datos para validar
en la fase entrenamiento
Valoutput=OutputTrain(posicionesval,:)';
%-------------------------------------------------------------------------
--
```

```
net= newff(minmax(Traininput),[75 1],{'logsig''purelin'},'trainrp');
%creamos la arquitectura red
VV.P=Valinput;  %asigna a la variable P de VV los datos de entrada
VV.T=Valoutput; %asigna a la variable T de VV los datos de salida

[net,tr,Y]=train(net,Traininput,Trainoutput,[],[],VV,[]); %entrenamos
usando datos entrenamiento y validación
net.trainParam  % nos da parámetros de la red entrenada

% y contiene los valores estimados de la red para los datos de
entrenamiento

% Gráfico de datos reales y estimaciones de la red neuronal
% usamos datos de entrenamiento
figure(1)
plot(Trainoutput,'b')
holdon
plot(Y,'r')
legend('Valores reales','Valores estimados Red')
title('Datos entrenamiento')
holdoff

MSEtrain=mean((Y-Trainoutput).^2); %mean square error entrenamiento

% ------------------------------------------------------------------------
%Se usa la red entrenada para el conjunto de datos no usados "Test"
TestRed=sim(net,InputTest')'; %estimación de la red para datos test

MSEtest=mean((TestRed-OutputTest).^2); %mean square error en datos test

figure(2)
plot(OutputTest(:,1),'b')
holdon
plot(TestRed(:,1),'r')
legend('Valores reales','Valores estimados Red')
title('Datos Test')
holdoff

[MSEtrain MSEtest]
[TestRed OutputTest]
```

- **Red empleada para la estimación del área de la fisura.**

```
clearall

datostrain=load('DatosTraccionTrain.txt'); %lee el fichero con los datos
datostest=load('DatosTraccionTest.txt');
% el fichero DatosTraccionTrain será usado para la fase de entrenamiento
% el fichero DatosTraccionTest tiene datos no usados en el entrenamiento
y
% será usado para calcular el error
```

```
[m,K]=size(datostrain); %m será número de filas y K número de columnas
[f,K]=size(datostest);

vinput= 2; %defino qué columnas contienen las variables de entrada
voutput= 5;  % columna que contiene variable salida

InputTrain=datostrain(:,vinput);
OutputTrain=datostrain(:,voutput);
[m,p]=size(InputTrain); %m número de datos y p número de variables
entrada
[m,ov]=size(OutputTrain); % m número de datos y ov número de variables
salida

InputTest=datostest(:,vinput);
OutputTest=datostest(:,voutput);

%-------------------------------------------------------------------------
-
%Para entrenar en las redes BackProp se usa un test de validación durante
%el entrenamiento. Vamos a seleccionar aleatoriamente este conjunto

nval=40; %tamaño del set de validación
ltrain=m-nval;

posicionesval(1,1)=ceil(rand(1,1)*m); %selecciona aleatoriamente la fila
que será el primer dato para set de validación
for i=2:nval
sumiguales=1;
while sumiguales>=1
posicionesval(i,1)=ceil(rand(1,1)*m);
iguales=zeros(i-1,1);
for j=i-1:-1:1
if posicionesval(i,1)==posicionesval(j,1)
iguales(j,1)=1;
else
iguales(j,1)=0;
end
end
sumiguales=sum(iguales);
end
end
posicionesval=posicionesval; %filas elegidas como datos validación

conta=0;
for i=1:m
    u=(posicionesval==i);
    u1=sum(u);
if u1>=1;
else
conta=conta+1;
posicionestrain(conta,1)=i;
end
end
posicionestrain=sort(posicionestrain); % filas para entrenamiento

% ------------------------------------------------------------------------
--
```

```
Traininput=InputTrain(posicionestrain,:)';  % estos son los datos para
entrenar, matriz dimensión p x ltrain
Trainoutput=OutputTrain(posicionestrain,:)'; % estos son los datos para
entrenar, matriz dimensión ov x ltrain

Valinput=InputTrain(posicionesval,:)'; %estos son los datos para validar
en la fase entrenamiento
Valoutput=OutputTrain(posicionesval,:)';
%-------------------------------------------------------------------------
--

net= newff(minmax(Traininput),[55 1],{'logsig''purelin'},'trainrp');
%creamos la arquitectura red
VV.P=Valinput;  %asigna a la variable P de VV los datos de entrada
VV.T=Valoutput; %asigna a la variable T de VV los datos de salida

[net,tr,Y]=train(net,Traininput,Trainoutput,[],[],VV,[]); %entrenamos
usando datos entrenamiento y validación
net.trainParam  % nos da parámetros de la red entrenada

% y contiene los valores estimados de la red para los datos de
entrenamiento

% Gráfico de datos reales y estimaciones de la red neuronal
% usamos datos de entrenamiento
figure(1)
plot(Trainoutput,'b')
holdon
plot(Y,'r')
legend('Valores reales','Valores estimados Red')
title('Datos entrenamiento')
holdoff

MSEtrain=mean((Y-Trainoutput).^2); %mean square error entrenamiento

% -------------------------------------------------------------------------
%Se usa la red entrenada para el conjunto de datos no usados "Test"
TestRed=sim(net,InputTest')'; %estimación de la red para datos test

MSEtest=mean((TestRed-OutputTest).^2); %mean square error en datos test

figure(2)
plot(OutputTest(:,1),'b')
holdon
plot(TestRed(:,1),'r')
legend('Valores reales','Valores estimados Red')
title('Datos Test')
holdoff

[MSEtrain MSEtest]
[TestRed OutputTest]
```

ii. Flexión

- **Red empleada para la estimación de la longitud característica de la fisura (α).**

```
clearall

datostrain=load('DatosFlexionTrain.txt'); %lee el fichero con los datos
datostest=load('DatosFlexionTest.txt');
% el fichero DatosFlexionTrain será usado para la fase de entrenamiento
% el fichero DatosFlexionTest tiene datos no usados en el entrenamiento y
% será usado para calcular el error

[m,K]=size(datostrain); %m será número de filas y K número de columnas
[f,K]=size(datostest);

vinput= 2; %defino qué columnas contienen las variables de entrada
voutput= 3;  % columna que contiene variable salida

InputTrain=datostrain(:,vinput);
OutputTrain=datostrain(:,voutput);
[m,p]=size(InputTrain); %m número de datos y p número de variables
entrada
[m,ov]=size(OutputTrain); % m número de datos y ov número de variables
salida

InputTest=datostest(:,vinput);
OutputTest=datostest(:,voutput);

%-------------------------------------------------------------------------
-
%Para entrenar en las redes BackProp se usa un test de validación durante
%el entrenamiento. Vamos a seleccionar aleatoriamente este conjunto

nval=40; %tamaño del set de validación
ltrain=m-nval;

posicionesval(1,1)=ceil(rand(1,1)*m); %selecciona aleatoriamente la fila
que será el primer dato para set de validación
for i=2:nval
sumiguales=1;
while sumiguales>=1
posicionesval(i,1)=ceil(rand(1,1)*m);
iguales=zeros(i-1,1);
for j=i-1:-1:1
if posicionesval(i,1)==posicionesval(j,1)
iguales(j,1)=1;
else
iguales(j,1)=0;
end
end
sumiguales=sum(iguales);
end
end
```

```
posicionesval=posicionesval; %filas elegidas como datos validación

conta=0;
for i=1:m
    u=(posicionesval==i);
    u1=sum(u);
if u1>=1;
else
conta=conta+1;
posicionestrain(conta,1)=i;
end
end
posicionestrain=sort(posicionestrain); % filas para entrenamiento

% ------------------------------------------------------------------------
--
Traininput=InputTrain(posicionestrain,:)';  % estos son los datos para
entrenar, matriz dimensión p x ltrain
Trainoutput=OutputTrain(posicionestrain,:)'; % estos son los datos para
entrenar, matriz dimensión ov x ltrain

Valinput=InputTrain(posicionesval,:)'; %estos son los datos para validar
en la fase entrenamiento
Valoutput=OutputTrain(posicionesval,:)';
%------------------------------------------------------------------------
--

net= newff(minmax(Traininput),[50 1],{'logsig''purelin'},'trainrp');
%creamos la arquitectura red
VV.P=Valinput;  %asigna a la variable P de VV los datos de entrada
VV.T=Valoutput; %asigna a la variable T de VV los datos de salida

[net,tr,Y]=train(net,Traininput,Trainoutput,[],[],VV,[]); %entrenamos
usando datos entrenamiento y validación
net.trainParam  % nos da parámetros de la red entrenada

% y contiene los valores estimados de la red para los datos de
entrenamiento

% Gráfico de datos reales y estimaciones de la red neuronal
% usamos datos de entrenamiento
figure(1)
plot(Trainoutput,'b')
holdon
plot(Y,'r')
legend('Valores reales','Valores estimados Red')
title('Datos entrenamiento')
holdoff

MSEtrain=mean((Y-Trainoutput).^2); %mean square error entrenamiento

% ------------------------------------------------------------------------
%Se usa la red entrenada para el conjunto de datos no usados "Test"
TestRed=sim(net,InputTest')'; %estimación de la red para datos test

MSEtest=mean((TestRed-OutputTest).^2); %mean square error en datos test
```

```
figure(2)
plot(OutputTest(:,1),'b')
holdon
plot(TestRed(:,1),'r')
legend('Valores reales','Valores estimados Red')
title('Datos Test')
holdoff

[MSEtrain MSEtest]
[TestRed OutputTest]
```

- **Red empleada para la estimación del factor de forma de la fisura (β).**

```
clearall

datostrain=load('DatosFlexionTrain.txt'); %lee el fichero con los datos
datostest=load('DatosFlexionTest.txt');
% el fichero DatosFlexionTrain será usado para la fase de entrenamiento
% el fichero DatosFlexionTest tiene datos no usados en el entrenamiento y
% será usado para calcular el error

[m,K]=size(datostrain); %m será número de filas y K número de columnas
[f,K]=size(datostest);

vinput= 2; %defino qué columnas contienen las variables de entrada
voutput= 4;  % columna que contiene variable salida

InputTrain=datostrain(:,vinput);
OutputTrain=datostrain(:,voutput);
[m,p]=size(InputTrain); %m número de datos y p número de variables
entrada
[m,ov]=size(OutputTrain); % m número de datos y ov número de variables
salida

InputTest=datostest(:,vinput);
OutputTest=datostest(:,voutput);

%-------------------------------------------------------------------------
-
%Para entrenar en las redes BackProp se usa un test de validación durante
%el entrenamiento. Vamos a seleccionar aleatoriamente este conjunto

nval=45; %tamaño del set de validación
ltrain=m-nval;

posicionesval(1,1)=ceil(rand(1,1)*m); %selecciona aleatoriamente la fila
que será el primer dato para set de validación
for i=2:nval
sumiguales=1;
while sumiguales>=1
posicionesval(i,1)=ceil(rand(1,1)*m);
iguales=zeros(i-1,1);
```

```
for j=i-1:-1:1
if posicionesval(i,1)==posicionesval(j,1)
iguales(j,1)=1;
else
iguales(j,1)=0;
end
end
sumiguales=sum(iguales);
end
end
posicionesval=posicionesval; %filas elegidas como datos validación

conta=0;
for i=1:m
    u=(posicionesval==i);
    u1=sum(u);
if u1>=1;
else
conta=conta+1;
posicionestrain(conta,1)=i;
end
end
posicionestrain=sort(posicionestrain); % filas para entrenamiento

% ------------------------------------------------------------------------
--
Traininput=InputTrain(posicionestrain,:)';  % estos son los datos para
entrenar, matriz dimensión p x ltrain
Trainoutput=OutputTrain(posicionestrain,:)'; % estos son los datos para
entrenar, matriz dimensión ov x ltrain

Valinput=InputTrain(posicionesval,:)'; %estos son los datos para validar
en la fase entrenamiento
Valoutput=OutputTrain(posicionesval,:)';
%------------------------------------------------------------------------
--

net= newff(minmax(Traininput),[70 1],{'logsig''purelin'},'trainrp');
%creamos la arquitectura red
VV.P=Valinput;  %asigna a la variable P de VV los datos de entrada
VV.T=Valoutput; %asigna a la variable T de VV los datos de salida

[net,tr,Y]=train(net,Traininput,Trainoutput,[],[],VV,[]); %entrenamos
usando datos entrenamiento y validación
net.trainParam  % nos da parámetros de la red entrenada

% y contiene los valores estimados de la red para los datos de
entrenamiento

% Gráfico de datos reales y estimaciones de la red neuronal
% usamos datos de entrenamiento
figure(1)
plot(Trainoutput,'b')
holdon
plot(Y,'r')
legend('Valores reales','Valores estimados Red')
title('Datos entrenamiento')
holdoff
```

```
MSEtrain=mean((Y-Trainoutput).^2); %mean square error entrenamiento

% ------------------------------------------------------------------
%Se usa la red entrenada para el conjunto de datos no usados "Test"
TestRed=sim(net,InputTest')'; %estimación de la red para datos test

MSEtest=mean((TestRed-OutputTest).^2); %mean square error en datos test

figure(2)
plot(OutputTest(:,1),'b')
holdon
plot(TestRed(:,1),'r')
legend('Valores reales','Valores estimados Red')
title('Datos Test')
holdoff

[MSEtrain MSEtest]
[TestRed OutputTest]
```

- **Red empleada para la estimación del área de la fisura.**

```
clearall

datostrain=load('DatosFlexionTrain.txt'); %lee el fichero con los datos
datostest=load('DatosFlexionTest.txt');
% el fichero DatosFlexionTrain será usado para la fase de entrenamiento
% el fichero DatosFlexionTest tiene datos no usados en el entrenamiento y
% será usado para calcular el error

[m,K]=size(datostrain); %m será número de filas y K número de columnas
[f,K]=size(datostest);

vinput= 2; %defino qué columnas contienen las variables de entrada
voutput= 5;  % columna que contiene variable salida

InputTrain=datostrain(:,vinput);
OutputTrain=datostrain(:,voutput);
[m,p]=size(InputTrain); %m número de datos y p número de variables
entrada
[m,ov]=size(OutputTrain); % m número de datos y ov número de variables
salida

InputTest=datostest(:,vinput);
OutputTest=datostest(:,voutput);

%-------------------------------------------------------------------
-
%Para entrenar en las redes BackProp se usa un test de validación durante
%el entrenamiento. Vamos a seleccionar aleatoriamente este conjunto
```

```
nval=40; %tamaño del set de validación
ltrain=m-nval;

posicionesval(1,1)=ceil(rand(1,1)*m); %selecciona aleatoriamente la fila
que será el primer dato para set de validación
for i=2:nval
sumiguales=1;
while sumiguales>=1
posicionesval(i,1)=ceil(rand(1,1)*m);
iguales=zeros(i-1,1);
for j=i-1:-1:1
if posicionesval(i,1)==posicionesval(j,1)
iguales(j,1)=1;
else
iguales(j,1)=0;
end
end
sumiguales=sum(iguales);
end
end
posicionesval=posicionesval; %filas elegidas como datos validación

conta=0;
for i=1:m
    u=(posicionesval==i);
    u1=sum(u);
if u1>=1;
else
conta=conta+1;
posicionestrain(conta,1)=i;
end
end
posicionestrain=sort(posicionestrain); % filas para entrenamiento

% ------------------------------------------------------------------------
--
Traininput=InputTrain(posicionestrain,:)';  % estos son los datos para
entrenar, matriz dimensión p x ltrain
Trainoutput=OutputTrain(posicionestrain,:)'; % estos son los datos para
entrenar, matriz dimensión ov x ltrain

Valinput=InputTrain(posicionesval,:)'; %estos son los datos para validar
en la fase entrenamiento
Valoutput=OutputTrain(posicionesval,:)';
%------------------------------------------------------------------------
--

net= newff(minmax(Traininput),[50 1],{'logsig''purelin'},'trainrp');
%creamos la arquitectura red
VV.P=Valinput;  %asigna a la variable P de VV los datos de entrada
VV.T=Valoutput; %asigna a la variable T de VV los datos de salida

[net,tr,Y]=train(net,Traininput,Trainoutput,[],[],VV,[]); %entrenamos
usando datos entrenamiento y validación
net.trainParam  % nos da parámetros de la red entrenada

% y contiene los valores estimados de la red para los datos de
entrenamiento
```

```
% Gráfico de datos reales y estimaciones de la red neuronal
% usamos datos de entrenamiento
figure(1)
plot(Trainoutput,'b')
holdon
plot(Y,'r')
legend('Valores reales','Valores estimados Red')
title('Datos entrenamiento')
holdoff

MSEtrain=mean((Y-Trainoutput).^2); %mean square error entrenamiento

% ------------------------------------------------------------------
%Se usa la red entrenada para el conjunto de datos no usados "Test"
TestRed=sim(net,InputTest')'; %estimación de la red para datos test

MSEtest=mean((TestRed-OutputTest).^2); %mean square error en datos test

figure(2)
plot(OutputTest(:,1),'b')
holdon
plot(TestRed(:,1),'r')
legend('Valores reales','Valores estimados Red')
title('Datos Test')
holdoff

[MSEtrain MSEtest]
[TestRed OutputTest]
```

iii. Flexo-tracción

- **Red empleada para la estimación de la longitud característica de la fisura(α).**

```
clearall

datostrain=load('DatosTracFlexTrain.txt'); %lee el fichero con los datos
datostest=load('DatosTracFlexTest.txt');
% el fichero DatosTraccionTrain será usado para la fase de entrenamiento
% el fichero DatosTraccionTest tiene datos no usados en el entrenamiento
y
% será usado para calcular el error

[m,K]=size(datostrain); %m será número de filas y K número de columnas
[f,K]=size(datostest);

vinput= 1, 2; %defino qué columnas contienen las variables de entrada
voutput= 3;  % columna que contiene variable salida

InputTrain=datostrain(:,vinput);
OutputTrain=datostrain(:,voutput);
[m,p]=size(InputTrain); %m número de datos y p número de variables
entrada
[m,ov]=size(OutputTrain); % m número de datos y ov número de variables
salida

InputTest=datostest(:,vinput);
OutputTest=datostest(:,voutput);

%-------------------------------------------------------------------------
-
%Para entrenar en las redes BackProp se usa un test de validación durante
%el entrenamiento. Vamos a seleccionar aleatoriamente este conjunto

nval=30; %tamaño del set de validación
ltrain=m-nval;

posicionesval(1,1)=ceil(rand(1,1)*m); %selecciona aleatoriamente la fila
que será el primer dato para set de validación
for i=2:nval
sumiguales=1;
while sumiguales>=1
posicionesval(i,1)=ceil(rand(1,1)*m);
iguales=zeros(i-1,1);
for j=i-1:-1:1
if posicionesval(i,1)==posicionesval(j,1)
iguales(j,1)=1;
else
iguales(j,1)=0;
end
end
sumiguales=sum(iguales);
end
end
posicionesval=posicionesval; %filas elegidas como datos validación
```

```
conta=0;
for i=1:m
    u=(posicionesval==i);
    u1=sum(u);
if u1>=1;
else
conta=conta+1;
posicionestrain(conta,1)=i;
end
end
posicionestrain=sort(posicionestrain); % filas para entrenamiento

% ------------------------------------------------------------------------
--
Traininput=InputTrain(posicionestrain,:)';  % estos son los datos para
entrenar, matriz dimensión p x ltrain
Trainoutput=OutputTrain(posicionestrain,:)'; % estos son los datos para
entrenar, matriz dimensión ov x ltrain

Valinput=InputTrain(posicionesval,:)'; %estos son los datos para validar
en la fase entrenamiento
Valoutput=OutputTrain(posicionesval,:)';
%------------------------------------------------------------------------
--

net= newff(minmax(Traininput),[70 1],{'logsig''purelin'},'trainrp');
%creamos la arquitectura red
VV.P=Valinput;  %asigna a la variable P de VV los datos de entrada
VV.T=Valoutput; %asigna a la variable T de VV los datos de salida

[net,tr,Y]=train(net,Traininput,Trainoutput,[],[],VV,[]); %entrenamos
usando datos entrenamiento y validación
net.trainParam  % nos da parámetros de la red entrenada

% y contiene los valores estimados de la red para los datos de
entrenamiento

% Gráfico de datos reales y estimaciones de la red neuronal
% usamos datos de entrenamiento
figure(1)
plot(Trainoutput,'b')
holdon
plot(Y,'r')
legend('Valores reales','Valores estimados Red')
title('Datos entrenamiento')
holdoff

MSEtrain=mean((Y-Trainoutput).^2); %mean square error entrenamiento

% ------------------------------------------------------------------------
%Se usa la red entrenada para el conjunto de datos no usados "Test"
TestRed=sim(net,InputTest')'; %estimación de la red para datos test

MSEtest=mean((TestRed-OutputTest).^2); %mean square error en datos test

figure(2)
```

```
plot(OutputTest(:,1),'b')
holdon
plot(TestRed(:,1),'r')
legend('Valores reales','Valores estimados Red')
title('Datos Test')
holdoff

[MSEtrain MSEtest]
[TestRed OutputTest]
```

- **Red empleada para la estimación de factor de forma de la fisura (β).**

```
clearall

datostrain=load('DatosTracFlexTrain.txt'); %lee el fichero con los datos
datostest=load('DatosTracFlexTest.txt');
% el fichero DatosTraccionTrain será usado para la fase de entrenamiento
% el fichero DatosTraccionTest tiene datos no usados en el entrenamiento
y
% será usado para calcular el error

[m,K]=size(datostrain); %m será número de filas y K número de columnas
[f,K]=size(datostest);

vinput= 1, 2; %defino qué columnas contienen las variables de entrada
voutput= 4;  % columna que contiene variable salida

InputTrain=datostrain(:,vinput);
OutputTrain=datostrain(:,voutput);
[m,p]=size(InputTrain); %m número de datos y p número de variables
entrada
[m,ov]=size(OutputTrain); % m número de datos y ov número de variables
salida

InputTest=datostest(:,vinput);
OutputTest=datostest(:,voutput);

%-------------------------------------------------------------------------
-
%Para entrenar en las redes BackProp se usa un test de validación durante
%el entrenamiento. Vamos a seleccionar aleatoriamente este conjunto

nval=50; %tamaño del set de validación
ltrain=m-nval;

posicionesval(1,1)=ceil(rand(1,1)*m); %selecciona aleatoriamente la fila
que será el primer dato para set de validación
for i=2:nval
sumiguales=1;
while sumiguales>=1
posicionesval(i,1)=ceil(rand(1,1)*m);
iguales=zeros(i-1,1);
```

```
for j=i-1:-1:1
if posicionesval(i,1)==posicionesval(j,1)
iguales(j,1)=1;
else
iguales(j,1)=0;
end
end
sumiguales=sum(iguales);
end
end
posicionesval=posicionesval; %filas elegidas como datos validación

conta=0;
for i=1:m
    u=(posicionesval==i);
    u1=sum(u);
if u1>=1;
else
conta=conta+1;
posicionestrain(conta,1)=i;
end
end
posicionestrain=sort(posicionestrain); % filas para entrenamiento

% ------------------------------------------------------------------------
--
Traininput=InputTrain(posicionestrain,:)';  % estos son los datos para
entrenar, matriz dimensión p x ltrain
Trainoutput=OutputTrain(posicionestrain,:)'; % estos son los datos para
entrenar, matriz dimensión ov x ltrain

Valinput=InputTrain(posicionesval,:)'; %estos son los datos para validar
en la fase entrenamiento
Valoutput=OutputTrain(posicionesval,:)';
%------------------------------------------------------------------------
--

net= newff(minmax(Traininput),[60 1],{'logsig''purelin'},'trainrp');
%creamos la arquitectura red
VV.P=Valinput;  %asigna a la variable P de VV los datos de entrada
VV.T=Valoutput; %asigna a la variable T de VV los datos de salida

[net,tr,Y]=train(net,Traininput,Trainoutput,[],[],VV,[]); %entrenamos
usando datos entrenamiento y validación
net.trainParam  % nos da parámetros de la red entrenada

% y contiene los valores estimados de la red para los datos de
entrenamiento

% Gráfico de datos reales y estimaciones de la red neuronal
% usamos datos de entrenamiento
figure(1)
plot(Trainoutput,'b')
holdon
plot(Y,'r')
legend('Valores reales','Valores estimados Red')
title('Datos entrenamiento')
holdoff
```

```
MSEtrain=mean((Y-Trainoutput).^2); %mean square error entrenamiento

% ------------------------------------------------------------------
%Se usa la red entrenada para el conjunto de datos no usados "Test"
TestRed=sim(net,InputTest')'; %estimación de la red para datos test

MSEtest=mean((TestRed-OutputTest).^2); %mean square error en datos test

figure(2)
plot(OutputTest(:,1),'b')
holdon
plot(TestRed(:,1),'r')
legend('Valores reales','Valores estimados Red')
title('Datos Test')
holdoff

[MSEtrain MSEtest]
[TestRed OutputTest]
```

- **Red empleada para la estimacióndel área de la fisura.**

```
clearall

datostrain=load('DatosTracFlexTrain.txt'); %lee el fichero con los datos
datostest=load('DatosTracFlexTest.txt');
% el fichero DatosTraccionTrain será usado para la fase de entrenamiento
% el fichero DatosTraccionTest tiene datos no usados en el entrenamiento
y
% será usado para calcular el error

[m,K]=size(datostrain); %m será número de filas y K número de columnas
[f,K]=size(datostest);

vinput=1, 2; %defino qué columnas contienen las variables de entrada
voutput= 5;  % columna que contiene variable salida

InputTrain=datostrain(:,vinput);
OutputTrain=datostrain(:,voutput);
[m,p]=size(InputTrain); %m número de datos y p número de variables
entrada
[m,ov]=size(OutputTrain); % m número de datos y ov número de variables
salida

InputTest=datostest(:,vinput);
OutputTest=datostest(:,voutput);

%-------------------------------------------------------------------
-
%Para entrenar en las redes BackProp se usa un test de validación durante
%el entrenamiento. Vamos a seleccionar aleatoriamente este conjunto
```

```
nval=40; %tamaño del set de validación
ltrain=m-nval;

posicionesval(1,1)=ceil(rand(1,1)*m); %selecciona aleatoriamente la fila
que será el primer dato para set de validación
for i=2:nval
sumiguales=1;
while sumiguales>=1
posicionesval(i,1)=ceil(rand(1,1)*m);
iguales=zeros(i-1,1);
for j=i-1:-1:1
if posicionesval(i,1)==posicionesval(j,1)
iguales(j,1)=1;
else
iguales(j,1)=0;
end
end
sumiguales=sum(iguales);
end
end
posicionesval=posicionesval; %filas elegidas como datos validación

conta=0;
for i=1:m
    u=(posicionesval==i);
    u1=sum(u);
if u1>=1;
else
conta=conta+1;
posicionestrain(conta,1)=i;
end
end
posicionestrain=sort(posicionestrain); % filas para entrenamiento

% ------------------------------------------------------------------------
--
Traininput=InputTrain(posicionestrain,:)';  % estos son los datos para
entrenar, matriz dimensión p x ltrain
Trainoutput=OutputTrain(posicionestrain,:)'; % estos son los datos para
entrenar, matriz dimensión ov x ltrain

Valinput=InputTrain(posicionesval,:)'; %estos son los datos para validar
en la fase entrenamiento
Valoutput=OutputTrain(posicionesval,:)';
%------------------------------------------------------------------------
--

net= newff(minmax(Traininput),[50 1],{'logsig''purelin'},'trainrp');
%creamos la arquitectura red
VV.P=Valinput;  %asigna a la variable P de VV los datos de entrada
VV.T=Valoutput; %asigna a la variable T de VV los datos de salida

[net,tr,Y]=train(net,Traininput,Trainoutput,[],[],VV,[]); %entrenamos
usando datos entrenamiento y validación
net.trainParam  % nos da parámetros de la red entrenada
```

```
% y contiene los valores estimados de la red para los datos de
entrenamiento

% Gráfico de datos reales y estimaciones de la red neuronal
% usamos datos de entrenamiento
figure(1)
plot(Trainoutput,'b')
holdon
plot(Y,'r')
legend('Valores reales','Valores estimados Red')
title('Datos entrenamiento')
holdoff

MSEtrain=mean((Y-Trainoutput).^2); %mean square error entrenamiento

% ------------------------------------------------------------------------
%Se usa la red entrenada para el conjunto de datos no usados "Test"
TestRed=sim(net,InputTest')'; %estimación de la red para datos test

MSEtest=mean((TestRed-OutputTest).^2); %mean square error en datos test

figure(2)
plot(OutputTest(:,1),'b')
holdon
plot(TestRed(:,1),'r')
legend('Valores reales','Valores estimados Red')
title('Datos Test')
holdoff

[MSEtrain MSEtest]
[TestRed OutputTest]
```